STAR (1+19)

NASA Conference Publication 2301

IPAD II

Advances in Distributed Data Base Management for CAD/CAM

Proceedings of a national symposium held in Denver, Colorado April 17-19, 1984

USN ITAB NASA

NASA Conference Publication 2301

IPAD II

Advances in Distributed Data Base Management for CAD/CAM

Compiled by
Susan W. Bostic
Langley Research Center
Hampton, Virginia

Proceedings of a national symposium sponsored by the National Aeronautics and Space Administration, the U.S. Naval Material Command, and the Industry Technical Advisory Board and held in Denver, Colorado April 17-19, 1984

NASA
National Aeronautics and Space Administration

Scientific and Technical Information Branch

1984

PREFACE

To respond to the national needs for improved productivity in engineering design and manufacturing, a NASA/U.S. Navy supported joint industry/government project has been underway denoted Integrated Programs for Aerospace-Vehicle Design (IPAD). The IPAD project objective is to improve engineering productivity through better use of computer-aided design and manufacturing (CAD/CAM) technology. It focuses on development of technology and associated software for integrated company-wide management of engineering information. IPAD project research is carried out primarily through a contract to The Boeing Commercial Airplane Company under the guidance of an Industry Technical Advisory Board (ITAB) composed of representatives of major aerospace and computer companies. Results to date are believed useful to a broad segment of both aerospace and nonaerospace organizations concerned with computer-aided design and manufacturing technology. A first IPAD Symposium was held September 17-19, 1980, in Denver, CO, and the Proceedings are published in NASA CP-2143. This second IPAD Symposium, "Advances in Distributed Data Base Management for CAD/CAM," is cosponsored by NASA, the U.S. Naval Material Command, and ITAB. The objectives of this Symposium are as follows:

1. To provide a greater awareness of the critical need by U.S. industry for advancements in distributed CAD/CAM data management capability

2. To present industry experiences and current and planned research in distributed data base management

3. To summarize IPAD data management contributions and their impact on U.S. industry and computer hardware and software vendors.

Symposium papers focus on distributed data management technology issues critical to the engineering and manufacturing process. Papers include a description of the government role in CAD/CAM data management research, CAD/CAM data management approaches in industry, research toward a future CAD/CAM distributed data management system, approach to CAD/CAM data management by hardware and software vendors, and industry experiences and exploitation of IPAD and related data management technology.

This document contains the manuscripts of the presentations available at the time of publication. Use of trade names or names of manufacturers in this report does not constitute an official endorsement of such products or manufacturers, either expressed or implied, by the National Aeronautics and Space Administration, or by the U.S. Navy.

SYMPOSIUM STEERING COMMITTEE

Paul Anderson
Ford Motor Company

Susan W. Bostic
NASA Langley Research Center

David Dieterich
Martin Marietta Aerospace

Robert E. Fulton
NASA Langley Research Center

R. L. Henderson
General Electric Company

George Kaler
General Dynamics, Convair Div.

Pierre LaBerge
Univac Corporation

C. R. Lewis
General Motors Technical Center

Ralph E. Miller, Jr.
Boeing Commercial Airplane Co.

T. C. Moody
Cessna Aircraft, Pawnee Div.

Warren E. Swanson
Playa del Rey, CA

Howard Syder
Boeing Commercial Airplane Co.

Dan Tanabe
Boeing Commercial Airplane Co.

Don Taylor
Boeing Computer Services

CONTENTS

SESSION 4 - DISTRIBUTED DATA BASE TECHNOLOGY DEVELOPMENT IN EUROPE
Chairman: Ralph E. Miller, Jr.

SESSION 5 - HARDWARE AND SOFTWARE VENDORS' APPROACH TO CAD/CAM DATA MANAGEMENT
Chairman: B. T. Jones

SESSION 6 - INDUSTRY APPLICATION OF A RELATIONAL INFORMATION MANAGER: RIM
Chairman: David Dieterich

LUNCHEON KEYNOTE SPEAKER
Admiral B. R. Inman (U.S.N., Ret.)

SESSION 7 - ICAM STUDIES OF MANUFACTURING DATA MANAGEMENT
Chairman: George Salley

SESSION 8 - IMPLEMENTATION OF DISTRIBUTED DATA BASE MANAGEMENT SYSTEMS
Chairman: P. LaBerge

SESSION 9 - ISSUES IN DISTRIBUTED DATA BASE MANAGEMENT PANEL DISCUSSION

SESSION 10 - IMPACT OF DATA BASE TECHNOLOGY IN UNIVERSITIES
Chairman: Michael E. Smith

MANAGEMENT OF CAD/CAM INFORMATION - KEY TO IMPROVED MANUFACTURING PRODUCTIVITY

Robert E. Fulton
NASA Langley Research Center

Jack Brainin
David Taylor Naval Ship Research and Development Center

SUMMARY

A key element to improved industry productivity is effective management of CAD/CAM information. To stimulate advancements in this area, a joint NASA/Navy/Industry project designated Integrated Programs for Aerospace-Vehicle Design (IPAD) is underway with the goal of raising aerospace industry productivity through advancement of technology to integrate and manage information involved in the design and manufacturing process. The project complements traditional NASA/DOD research to develop aerospace design technology and the Air Force's Integrated Computer-Aided Manufacturing (ICAM) program to advance CAM technology. IPAD research is guided by an Industry Technical Advisory Board (ITAB) composed of over 100 representatives from aerospace and computer companies. This paper summarizes IPAD accomplishments to date in development of requirements and prototype software for various levels of company-wide CAD/CAM data management and discusses (1) plans for development of technology for management of distributed CAD/CAM data and (2) information required to control future knowledge-based CAD/CAM systems.

INTRODUCTION

For the United States to remain competitive in the world market, improvements in industrial productivity are essential. A key element to improved productivity is the advancement and effective use of computer-aided design/manufacturing (CAD/CAM) technology. To stimulate advancements in CAD/CAM technology, a joint NASA/Navy/Industry project, denoted Integrated Programs for Aerospace-Vehicle Design (IPAD), is underway and is making significant progress (fig. 1). The project goal: raise aerospace industry productivity through advancement of technology to integrate and manage information involved in the design and manufacturing process. The program complements traditional NASA/DOD research to develop aerospace design technology and the Air Force Integrated Computer-Aided Manufacturing (ICAM) program to advance CAM technology. Work under the IPAD project is being done principally through a prime contract to the Boeing Commercial Airplane Company under the guidance of an Industry Technical Advisory Board (ITAB) composed of members of aerospace and Navy contractors and computer companies (fig. 2). ITAB reviews provide a regular forum for over 100 engineering and computer organizations to hold indepth discussions of critical CAD/CAM issues which direct IPAD research and spur internal company efforts. This paper summarizes the background of the IPAD program, NASA and Navy needs for data management technology, an approach to developing CAD/CAM data management software, and priorities for future development.

BACKGROUND

In the late 1970's, NASA-sponsored work on the project showed that significant improvement in engineering productivity was possible through effective automation of information management tasks. The studies included indepth investigation of representative design processes, quantification of the flow of information through such processes, and determination of how automation could significantly aid that flow (fig. 3). Requirements and system design of a future integrated engineering information management system was developed and the concepts were subjected to extensive review by ITAB. It was concluded that available commercial computer software would not meet these requirements and that leadership was needed to stimulate development in critical areas (refs. 1-4).

While the Navy has been associated with IPAD since its inception, in 1982 the Manufacturing Technology Program of the Naval Material Command formally joined NASA to extend IPAD data management research to address the added requirements of manufacturing. Requirements for manufacturing data management have been developed from (1) earlier IPAD requirements studies (ref. 1); (2) ICAM-funded tasks on Product Definition Data Interface by McDonnell Aircraft Company (ref. 5) and Integrated Information Support System by General Electric Company (ref. 6); and (3) an IPAD-funded study of manufacturing information flow for a naval aircraft by Grumman Aerospace Corporation (ref. 7). An initial report on manufacturing data management requirements has been prepared for ITAB to guide future CAD/CAM research. References 1 and 8 provide comprehensive summaries of results from the IPAD program, as well as how organizations are using IPAD and related approaches to address CAD/CAM data management needs.

NASA/NAVY NEED FOR CAD/CAM DATA MANAGEMENT TECHNOLOGY

Advancement of technology to manage CAD/CAM information is an important activity for NASA/Navy leadership and has the potential for significant NASA/DOD benefit. For example, NASA responsibilities include development of key high-risk technologies to support both DOD and civilian aerospace industry needs, and many studies show improved CAD/CAM data management capability is critical to improve industry productivity (ref. 9). NASA also needs engineering data management capabilities to support internal research activities such as scientific computations, development of wind-tunnel models, development and operation of experimental facilities, and project troubleshooting (ref. 10). Furthermore, NASA and its contractors need extensive CAD/CAM data management capabilities to support cost-effective development of high-technology projects such as a future space station and/or other spacecraft (ref. 11).

Navy needs for CAD/CAM data management technology are many and typical of DOD (ref. 12). For example, technology addressed in the IPAD project has the potential to significantly reduce cost and improve productivity in design and manufacture of Navy airplanes, weapons, and ships. The IPAD development strategy, which is based on aerospace industry requirements, appears applicable to many engineering systems and should be useful to the entire spectrum of Navy design and construction. The importance of data management to Navy ship development is illustrated in the following paragraphs.

The Navy ship acquisition process is divided into two stages bound together by the need for common data transfer. Early stage design, performed by the Naval Sea Systems Command, consists of feasibility studies, preliminary design, and contract design. This work includes cost and performance trade-offs, platform sizing and definition, and specification generation. The end product is a contract bidder's package which includes the development specifications for the ship's detail design. The second stage, performed by private shipbuilders, deals with the ship detail design and construction process. This second stage accomplishes selection, procurement, and arrangement of ship components; final detailing of distributive systems; and production of working drawings, lists, and miscellaneous information needed for construction and testing (fig. 4). Detail design and construction includes functional systems for hull/structures, propulsion machinery, combat systems, and distributive systems. Detail design development strategy must accommodate naval shipbuilders who use a variety of computer hardware for technical tasks. Portability is a key need for all engineering data bases, and maintenance of data integrity and consistency among different applications and different subcontractors greatly magnifies the complexity of the management of engineering data.

Ship design data base management requirements are similar to many other engineering requirements, but differ markedly from transaction-oriented business applications. Some significant differences are related to volume of data, volatility and complexity, variety of applications that use the data, and iterative nature of the design process. A complete ship design data base, for example, contains a digital description of ship components and their interrelationships. Data volume for detail design and construction is estimated at 2 billion data items; figure 5, for example, shows the evolution of design data for an aircraft carrier. Data base management systems to support this work must include flexible data modeling and multiple data structures. Convenient features are needed to update and retrieve data by name or through queries based on data values and conditions, to cross reference data in different files, and to obtain information on various data base attributes. Data items in the data base must be separated from the engineering design and analyses programs to negate the cumulative effect of continual changes to data items during design. Each design discipline must also have the capability of accessing the data from its own perspective. The data base must be capable of being accessed by multiple users concurrently, each using the data in different ways and for different purposes. At the same time, procedures must be provided to protect data from being accessed or modified by unauthorized persons. Thus, the CAD/CAM data base will continually change and grow throughout the design cycle due to the iterative nature of the ship design process, and effective computer software is needed to support that process.

The critical need for data management in a NASA or Navy engineering/manufacturing organization stems from the complexity and volume of data and the large number of activities requiring the data. The lack of a unified approach to data base management can result in inefficient storage, control, and use of data. Data redundancy can become widespread and can result in significant potential for errors due to lack of data integrity or to attendant complexity in configuration management procedures.

CAD/CAM DATA MANAGEMENT DEVELOPMENT APPROACH

Under the guidance of ITAB, the IPAD project has developed prototype computer software to meet many CAD/CAM information management requirements (refs. 3 and 4). Some of the basic requirements driving CAD/CAM systems development (refs. 13-21) include (fig. 6): (1) accommodate many different views of data from a variety of users and computing storage devices; (2) allow many levels of data descriptions to support a wide variety of engineering organizations and tasks; (3) permit easy changes in data definition as work progresses; (4) allow data to be distributed over networks of computers of various manufacture; (5) permit data definitions to be readily extended as needs arise; (6) store and manipulate geometry information; (7) embody adequate configuration management features; and (8) provide extensive capability to management information describing stored data. The IPAD approach taken is to conduct appropriate research and develop prototype software for a future network of computers (refs. 22-24). To provide the required CAD/CAM functionality, and yet meet software performance requirements, data base management is staged at two or more levels with different software capabilities needed for both the local (user) level and global (project) level (fig. 7). With such a tiered data base management approach, today's inconvenient file-oriented procedures (fig. 8) can be replaced by future procedures (fig. 9) where convenient user languages efficiently create, store, manipulate, access, and control information in accordance with CAD/CAM requirements.

Prototype software has now been developed under the IPAD project at both the local and global levels. A system denoted Relational Information Management (RIM) was developed for local level data management. RIM is based on the highly flexible relational models which organize and manage engineering and scientific information according to tables and relationships among tables. Its features include interactive queries, report writer, and FORTRAN interface. RIM was first operational in 1979 and is now a mature system. In 1981, it served as a critical information management capability to support NASA investigations (ref. 25) of the integrity of 30,000 tiles on the space shuttle (fig. 10). The success of RIM in such evaluations has led to its continued development and enhancement by government and industry. A public Version 5.0 is available from COSMIC* for CDC, IBM, DEC, UNIVAC, PRIME, and Harris computers. Commercial organizations have continued to enhance RIM and now provide compatible RIM derivative software (e.g., BCS/RIM and MicroRIM) and associated maintenance and support for such software operational on a wide range of computers (from personal computers to super computers). Commercial versions of RIM are being used extensively by industry (refs. 26-29), and one version has been adopted by the Naval Sea Systems Command for use in its early stage ship design integration process.

IPAD research has continued on development of a global data base management system denoted IPAD Information Processor (IPIP). The approach taken in IPIP is to provide the capability within one system to manage information composed of a wide variety of information structures including hierarchical, network, relational, and geometric. The IPIP approach uses multiple levels of information formats (schemata) to permit unlimited reorganization of information as work progresses (fig. 11).

*Computer Software Management and Information Center
112 Barrow Hall
University of Georgia
Athens, GA 30602

Each schema is connected to other schemata via a general-purpose mapping capability (language). IPIP is a new concept, still in test and evaluation phases, and is currently operational on a CDC computer. Its approach to management of geometric data is a unique and important concept which could be very important to future integration of design and manufacturing. A critical technical challenge to IPIP development has been to provide the high degree of engineering user flexibility and yet achieve acceptable response times. In late 1983, a test system which has user responses for test problems of less than 0.5 seconds was provided to selected ITAB organizations to support evaluations such as that illustrated in figure 12. IPIP has also been established by one computer vendor as an "as is" product and limited support is provided by the vendor for its installation and evaluation. IPAD results to date in defining CAD/CAM data management requirements and in developing prototype software have helped stimulate development of commercial CAD/CAM data management software (refs. 30-33), and several computer vendors plan release in 1984 of relational-type data management systems which address many of the CAD/CAM requirements identified in IPAD research. IPAD results have also helped stimulate infusion of data base management technology into university engineering research (refs. 34-36).

A critical CAD/CAM requirement not yet contained within any available or planned commercial data management system is the ability to efficiently manage geometry information in concert with other engineering data (fig. 13). Through use of the multischema capability, IPIP provides the first approach to management of geometry information within a data management system (refs. 13 and 14). The IPIP approach provides software capability to create on top of the basic geometric data an information structure having an unlimited number of geometric descriptions (schemata). One geometry schema includes the evolving geometry/graphics standard, Initial Graphics Exchange Specifications (IGES). This IPIP information structure concept opens the door for convenient integration of geometric information with other types of information associated with a CAD/CAM development process (fig. 14). An evaluation of the IPIP geometry concept is now underway, and comparisons are being made with other approaches in which management of the geometric data takes place oustide the basic data base manager.

PRIORITIES FOR FUTURE CAD/CAM DEVELOPMENT

A key data management requirement for CAD/CAM integration is the ability to manage unified information distributed across computers of different manufacture with the user flexibility provided by software such as RIM and IPIP. A typical company may have several different computers to support its combined engineering manufacturing activity; the addition of subcontractors introduces even more heterogeneity in computers. Examples of recent high-technology developments include the space shuttle (fig. 15) and the Navy advanced aircraft (e.g., fig. 16), wherein major components were developed by many widely dispersed companies, each having different computer complexes. IPAD research has begun development of technology for distributed data management. The basic IPIP design was planned for a distributed complex, and prototype software was developed in 1980 to provide high-speed (greater than 10^6 bits/sec) information transfer between CDC CYBER 730 and a DEC VAX 11/780. This transfer concept is a critical element to distributed data management and has already been expanded by vendors into commercial products (e.g., NETEX). IPAD research underway in 1984 is to investigate distributed data management across a test-bed system composed of a CDC CYBER 835, an IBM 4341, and a DEC VAX 11/780, with each computer utilizing

different data base management software (fig. 17). Critical needs identified by ITAB, NASA, and the Navy for CAD/CAM related data management research over the next few years include:

1. Refinement of manufacturing data management requirements

2. Development of executive software to control information management over a network of heterogeneous computers

3. Development of distributed data management software capabilities

4. Extension of geometry data management concepts to include solid modeling data

5. Development of multidisciplinary analysis/design data management approaches for sequential and concurrent processing computers (fig. 18)

6. Development of data management approaches for expert engineering systems (fig. 19)

A National Research Council report states (ref. 9): The use of computers in design and manufacturing offers the potential of an integrated information system that encompasses product planning, designing, manufactural engineering, purchasing, materials requirements planning, manufacturing, quality assurance, and customer acceptance. A single product definition data base containing an electronic description of the designed products that are being constructed or manufactured is a keystone to the successful utilization of CAD/CAM technology. The NASA/Navy IPAD project under the guidance of ITAB has helped focus unified government/industry technology development on this important national productivity need.

REFERENCES

1. IPAD: Integrated Programs for Aerospace-Vehicle Design. NASA CP-2143, 1980.

2. Fulton, Robert E.: Overview of Integrated Programs for Aerospace-Vehicle Design (IPAD). NASA TM-81874, 1980.

3. Fulton, Robert E.: CAD/CAM Approach to Improving Industry Productivity Gathers Momentum. Astronaut. & Aeronaut., vol. 20, no. 1, Feb. 1982, pp. 64-70.

4. Fulton, Robert E.: Critical Technology Issues for Integrated CAD/CAM. Proceedings of the Third Annual Conference and Exposition of the National Computer Graphics Association, Inc. - Volume 1. Nat. Computer Graphics Assoc., Inc., 1982, pp. 326-335.

5. Birchfield, Burt; and Downey, Peter: Product Definition Data Interface. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 20 of this compilation.)

6. DeJean, Jean Pierre: Integrated Information Support System (System Overview). General Electric Paper presented at IPAD II National Symposium, Denver, CO, Apr. 17-19, 1984.

7. Manufacturing-Data Usage at Grumman - A Study of the Fabrication Phase, Grumman Aerospace Report, prepared under Contract Y4D5803-0B3JN, June 1983.

8. IPAD II - Advances in Distributed Data Base Management for CAD/CAM. NASA CP-2301, 1984.

9. Comm. Navy Shipbuilding Technol., Marine Board: Productivity Improvements in U.S. Naval Shipbuilding. Contract N00024-82-C-5349, Comm. Eng. & Tech. Sys., Natl. Res. Counc., 1982, p. 32.

10. Colladay, R. S.: NASA Needs for Engineering/Scientific Data Management Technology. NASA paper presented at IPAD II National Symposium, Denver, CO, Apr. 17-19, 1984.

11. Dube, R. Peter; and Johnson, H. Randall: Computer-Assisted Engineering Data Base. Presented at ASME Winter Annual Meeting, Boston, MA, Nov. 13-18, 1983. ASME Paper 83-WA/Aero-11.

12. McInnis, John W.: The Role of CAD/CAM Data Management in Navy Manufacturing Technology. U.S. Navy paper presented at IPAD II National Symposium, Denver, CO, Apr. 17-19, 1984.

13. Shoosmith, J. N.; and Fulton, R. E. (Guest Editors): Geometric Modeling. Special Editions of Computer Graphics and Applications, Oct./Nov. 1983.

14. Dube, R. Peter; and Smith, Marcy Rivers: Managing Geometric Information with a Data Base Management System. Special Edition of Computer Graphics and Applications on Geometric Modeling, Oct. 1983, pp. 57-62.

15. Ford, S. Jeane: CIM's Bridge From CADD to CAM - Data Management Requirements for Manufacturing Engineering. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 4 of this compilation.)

16. Barnes, Robert D.: Technology for the Product & Process Data Base. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 5 of this compilation.)

17. Baur, R. G.; Donthnier, M. L.; Moran, M. C.; Mortman, I.; and Pinter, R. S.: Materials Properties Data Base Computerization. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 6 of this compilation.)

18. Bohn, Pierre: CAD/CAM and Scientific Data Management at Dassault. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 10 of this compilation.)

19. Neuhold, Erich J.: Distributed Data Base Systems With Special Emphasis Toward POREL. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 11 of this compilation.)

20. Piscitelli, H.: Data Base Design and Implementation in Multiplant and Multi-Company Corporations. Goodyear paper presented at IPAD II National Symposium, Denver, CO, Apr. 17-19, 1984.

21. Mitchell, Mary J. and Barkmeyer, Edward J.: Data Distribution in the NBS Automated Manufacturing Research Facility. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 23 of this compilation.)

22. McKenna, E. G.: Design Versus Manufacturing Data Base Management Requirements. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 7 of this compilation.)

23. Balza, R. M.; Beaudet, R. W.; and Johnson, H. R.: A Distributed Data Base Management Facility for the CAD/CAM Environment. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 8 of this compilation.)

24. Bryant, W. A.; and Smith, M. R.: Data Management for Computer-Aided Engineering (CAE). IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 9 of this compilation.)

25. Giles, Gary L.; and Vallas, Maria: Use of an Engineering Data Management System in the Analysis of Space Shuttle Orbiter Tiles. NASA TM-83215, 1981.

26. Breitbart, Yuri J.; and Hartweg, Larry R.: RIM as an Implementation Tool for a Distributed Heterogeneous Database. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 16 of this compilation.)

27. Karr, Patricia H.; and Wilson, David J.: RIM as the Data Base Management System for a Material Properties Data Base. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 17 of this compilation.)

28. Smith, Maurice: Integrating Heterogeneous Nongeometric Information Systems Using RIM. Bendix paper presented at IPAD II National Symposium, Denver, CO, Apr. 17-19, 1984.

29. Wray, William O., Jr.: A Wind Tunnel Database Using RIM. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 19 of this compilation.)

30. Feldman, Harley D.: Control Data ICEM - A Vendor's IPAD-Like System. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 12 of this compilation.)

31. Hartzband, D.: IPAD Influence on Digital Equipment Corporation. Digital Equipment paper presented at IPAD II National Symposium, Denver, CO, Apr. 17-19, 1984.

32. Gingerich, J. Z.: Future Developments of a Hybrid CAD/CAM System. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 14 of this compilation.)

33. Westervelt, Frank: High Performance Data Base Systems for CAD/CAM CIM Applications. SMC Technology paper presented at IPAD II National Symposium, Denver, CO, Apr. 17-19, 1984.

34. Plummer, Otho R.: The Database Management System: A Topic and a Tool. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 24 of this compilation.)

35. Fenves, Steven J.: The Role of DBMS in Design Research. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 25 of this compilation.)

36. Leach, Lanse M.; and Wozny, Michael J.: The Impact of IPAD on CAD/CAM Database University Research. IPAD II - Advances in Distributed Data Base Management for CAD/CAM, NASA CP-2301, 1984. (Paper 26 of this compilation.)

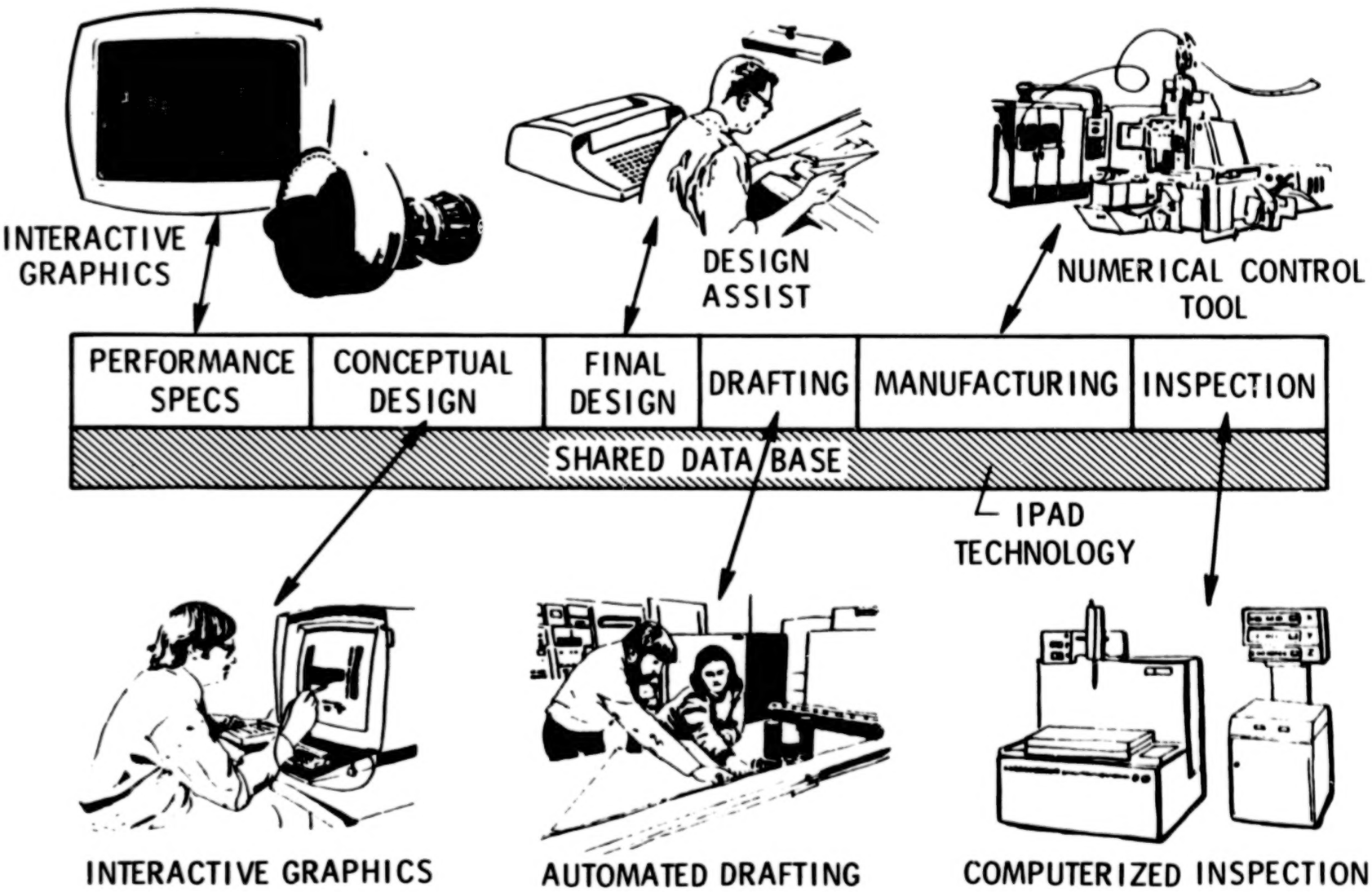
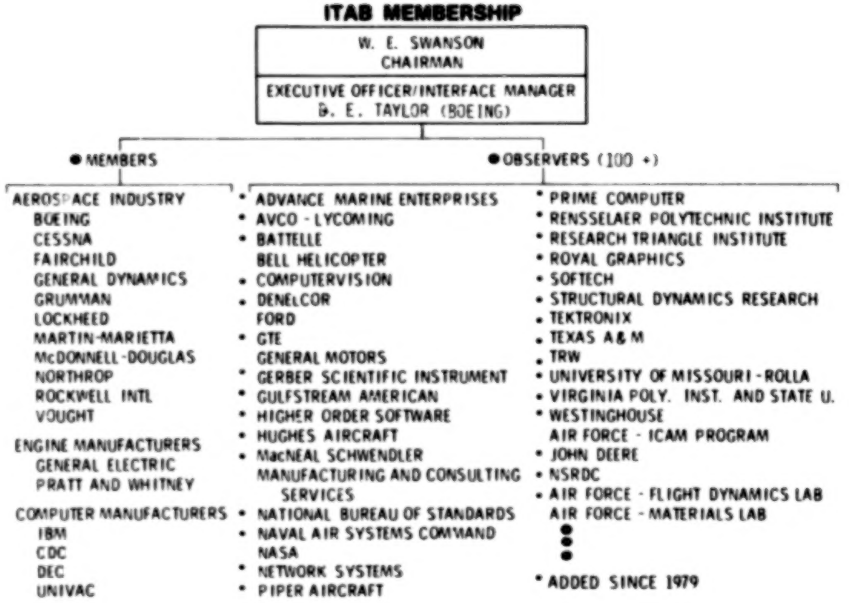


Figure 1.- Joint NASA/Navy research program to develop technology to manage CAD/CAM information.



ITAB ACTIVITIES

GUIDE DEVELOPMENT TO MEET INDUSTRY NEEDS

REVIEW/CRITIQUE ONGOING WORK

EVALUATE PROTOTYPE SOFTWARE

USE IPAD TECHNOLOGY AND PRODUCTS TO SPUR IN-HOUSE PLANNING AND DEVELOPMENT

Figure 2.- IPAD Industry Technical Advisory Board (ITAB).

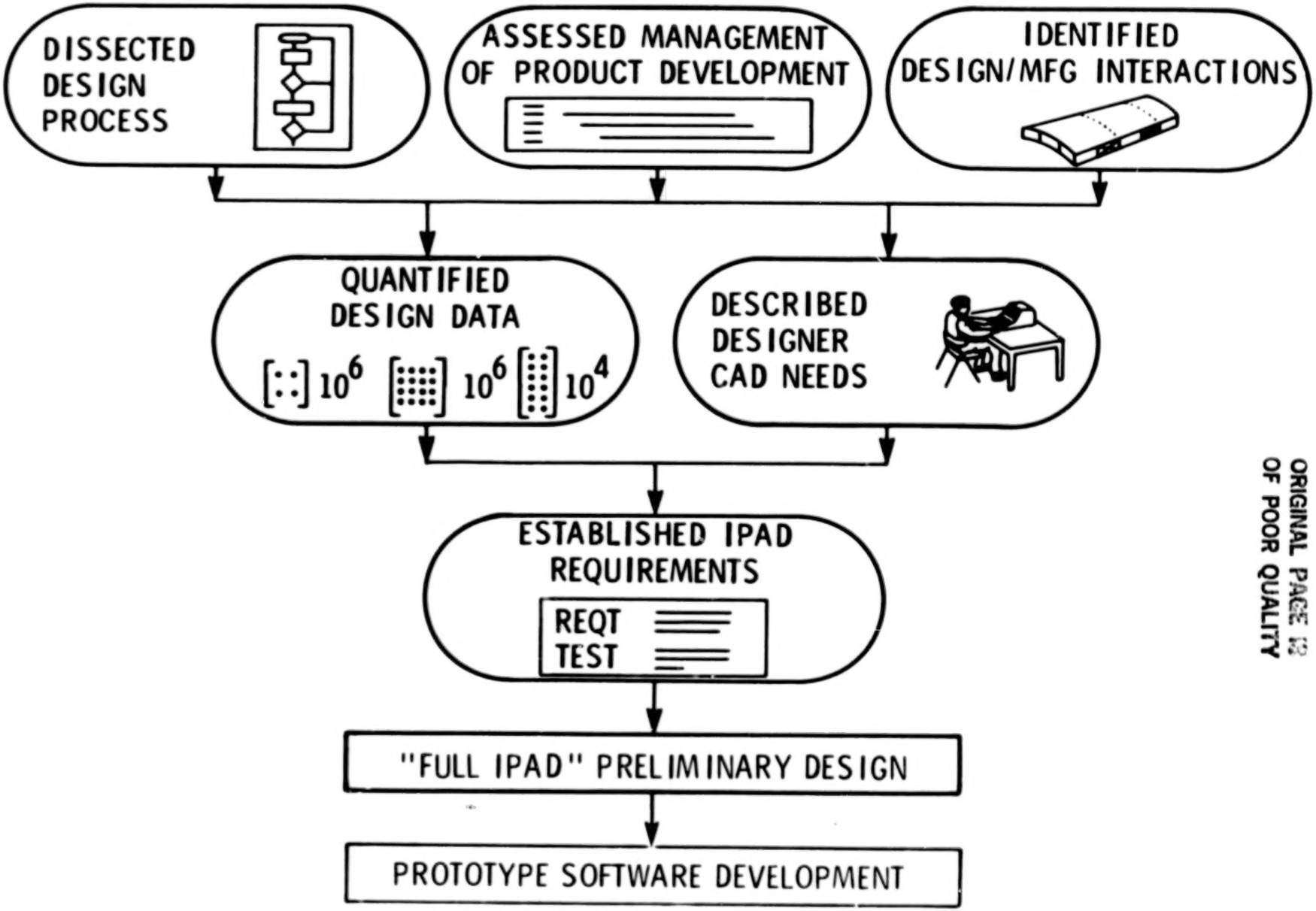


Figure 3.- IPAD approach to developing prototype software.

Figure 4.- Navy ship detail design system.

MISSION REQUIREMENTS -
18 PAGES

PRELIMINARY DESIGN -
12 PLANS AND 70 PAGE BOOKLET

CONTRACT DESIGN -
150 PLANS AND 800 PAGE SPEC.

DETAIL DESIGN -
7,000 PLANS AND 70 TECH. MANUAL

Figure 5.- Growth of information during ship design process. Data for aircraft carrier.

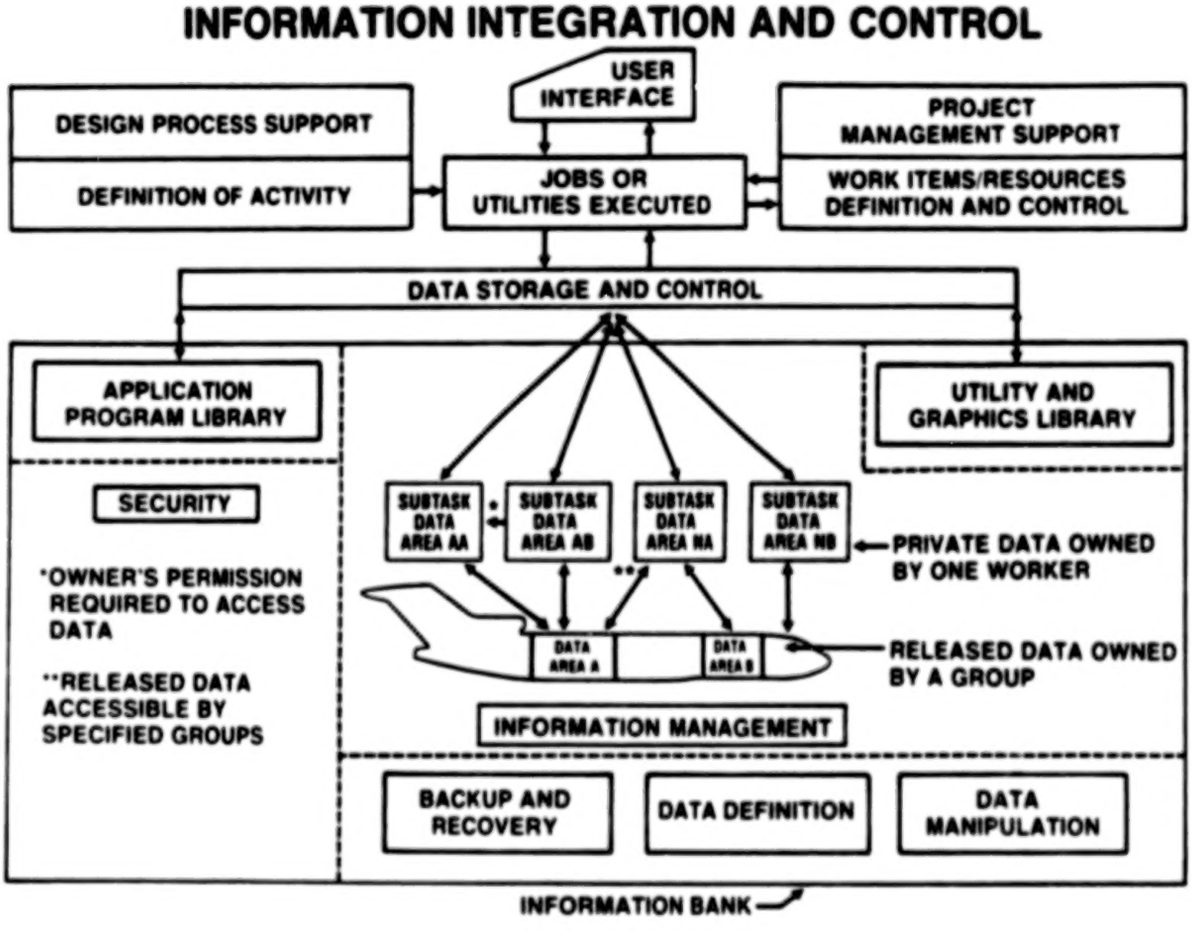


Figure 6.- Engineering use of integrated CAD/CAM data management system.

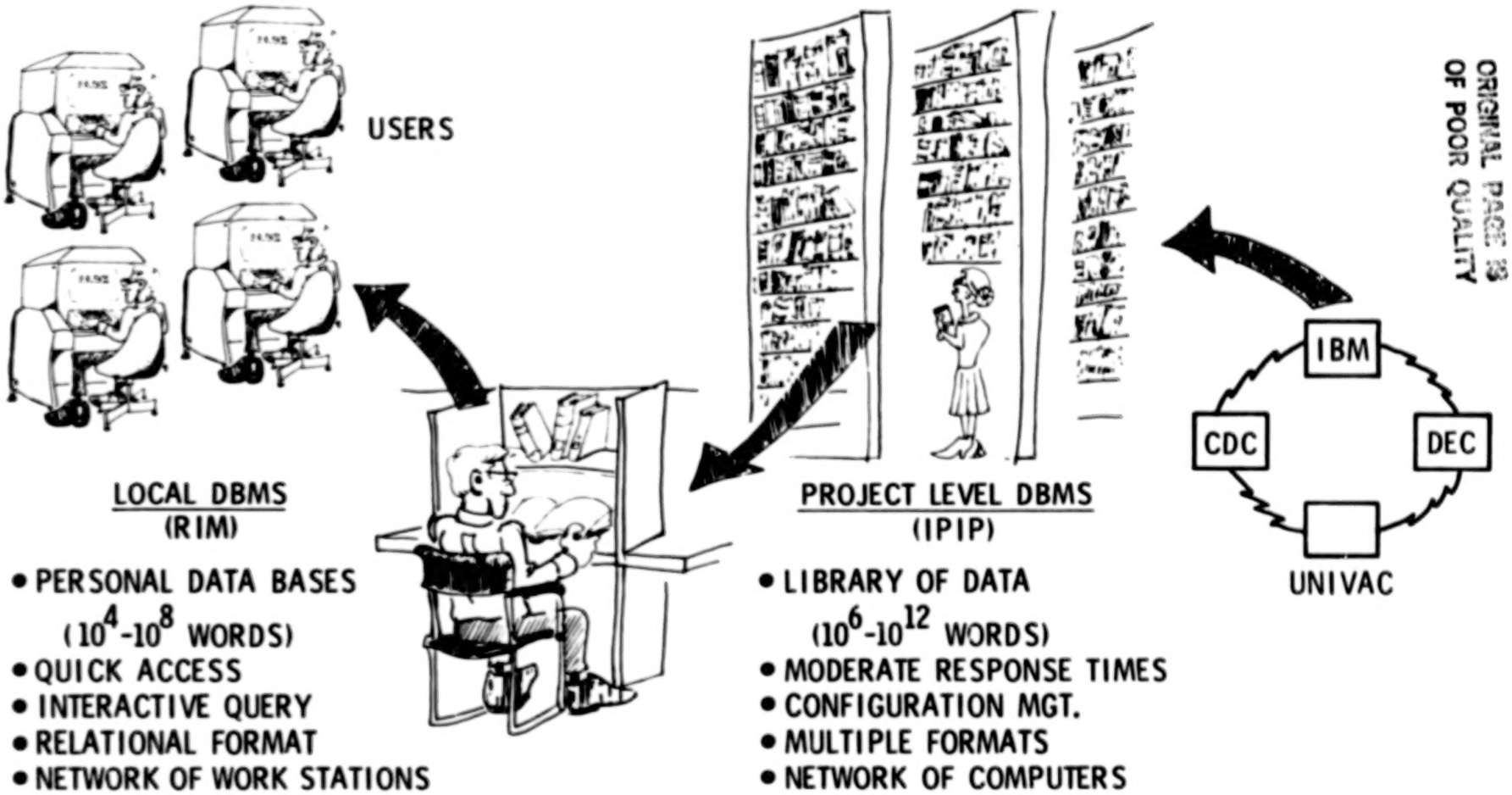


Figure 7.- IPAD multilevel approach to engineering data base management.

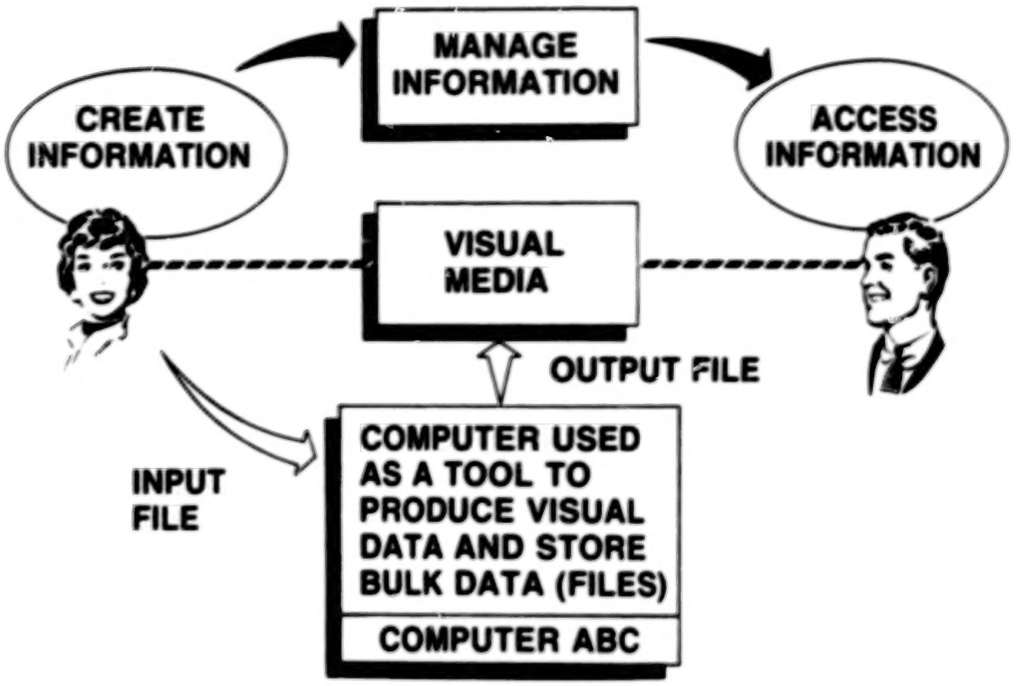
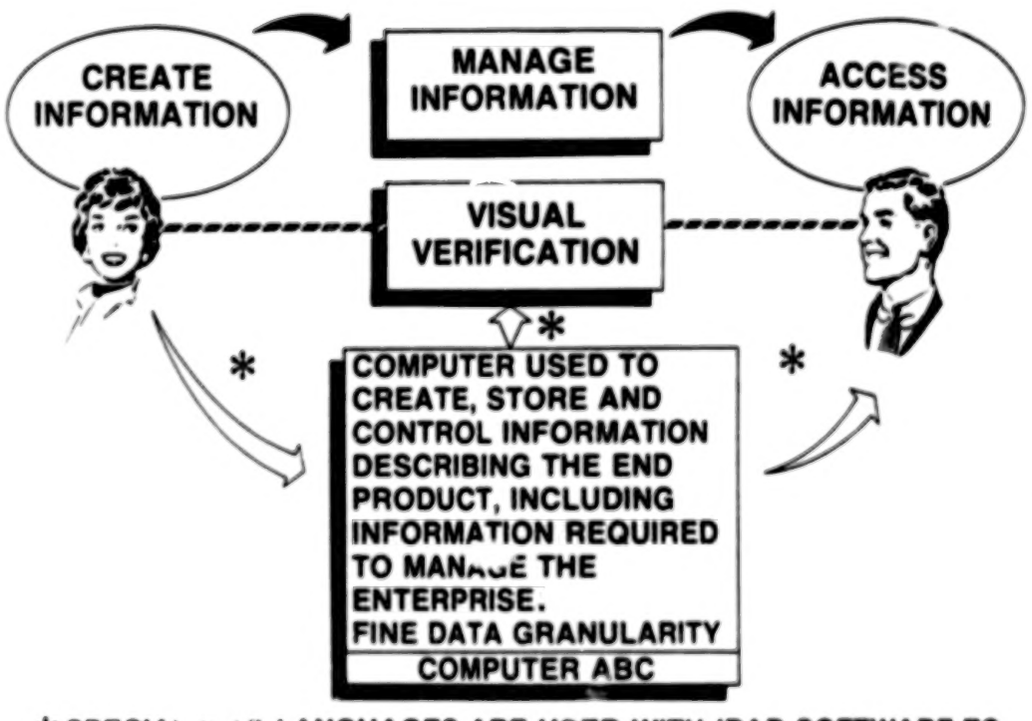


Figure 8.- Current file-oriented approach for CAD/CAM information.



*SPECIAL IPAD LANGUAGES ARE USED WITH IPAD SOFTWARE TO INTERFACE WITH THE COMPUTER-BASED DATA.

Figure 9.- Future data base management approach for CAD/CAM information.

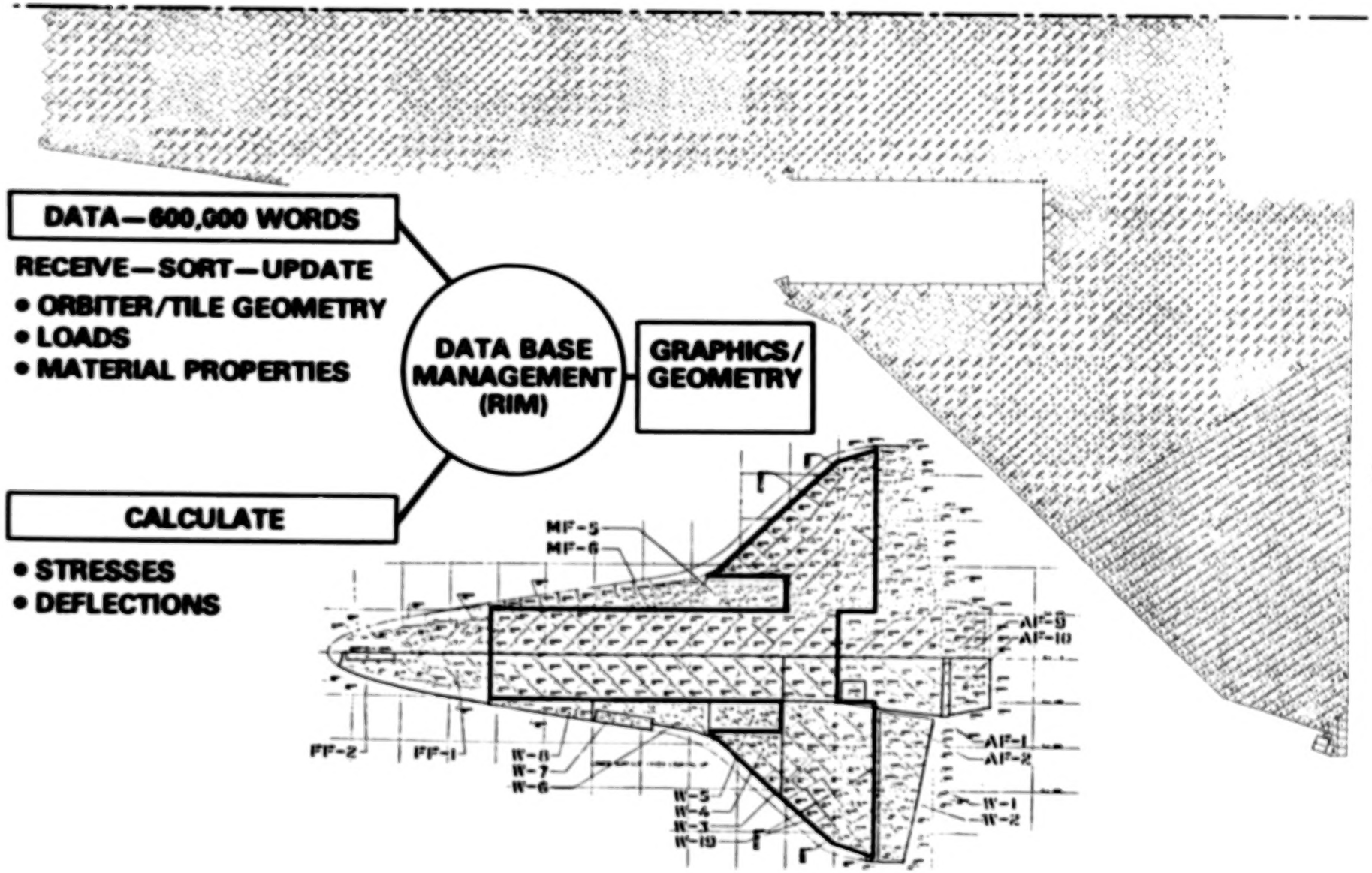


Figure 10.- Use of IPAD/RIM data manager to support investigation of space shuttle tile analyses.

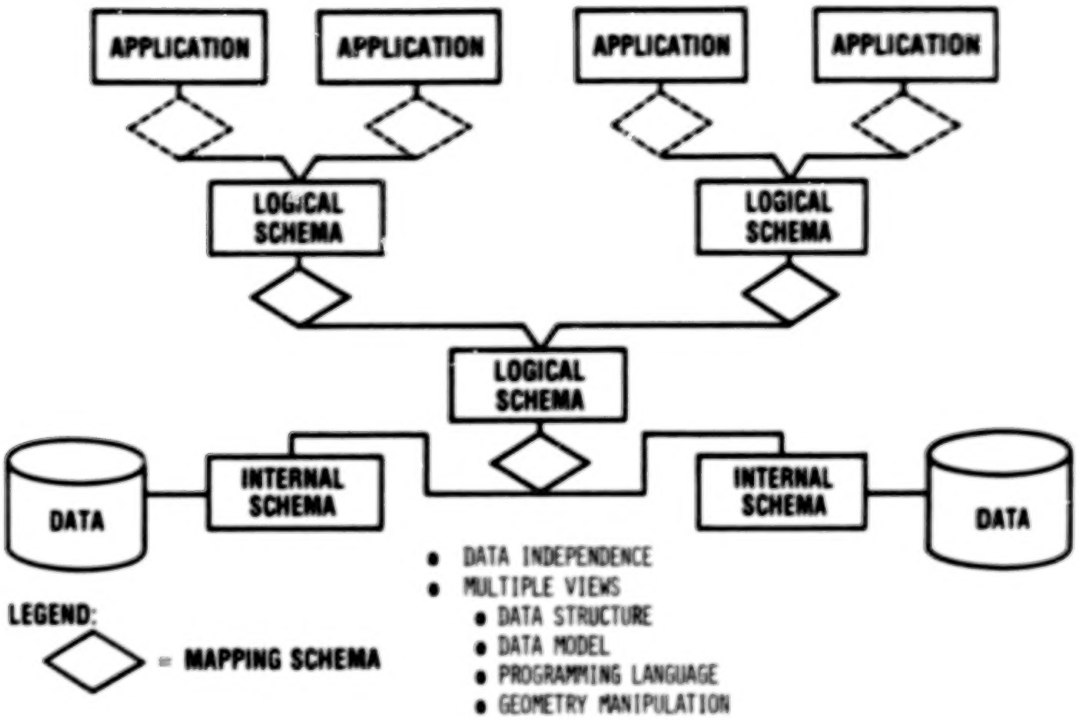


Figure 11.- Typical arrangement of IPIP data schemata (formats) to connect application schemata to storage schemata.

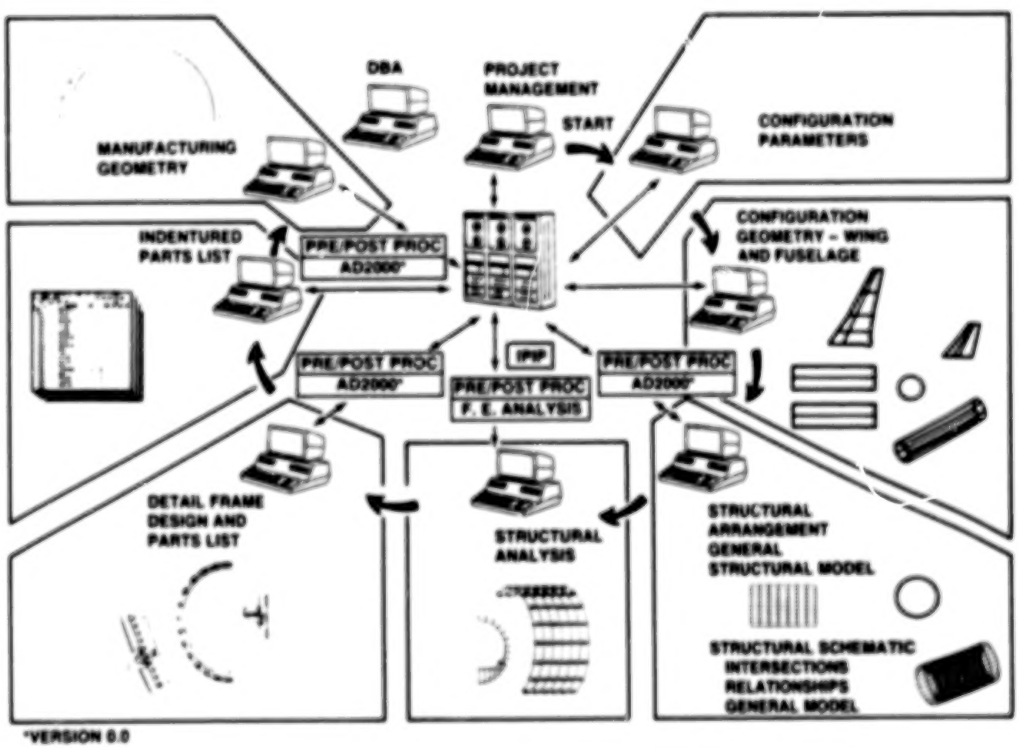


Figure 12.- Planned test demonstrations to evaluate IPAD/IPIP for CAD/CAM use.

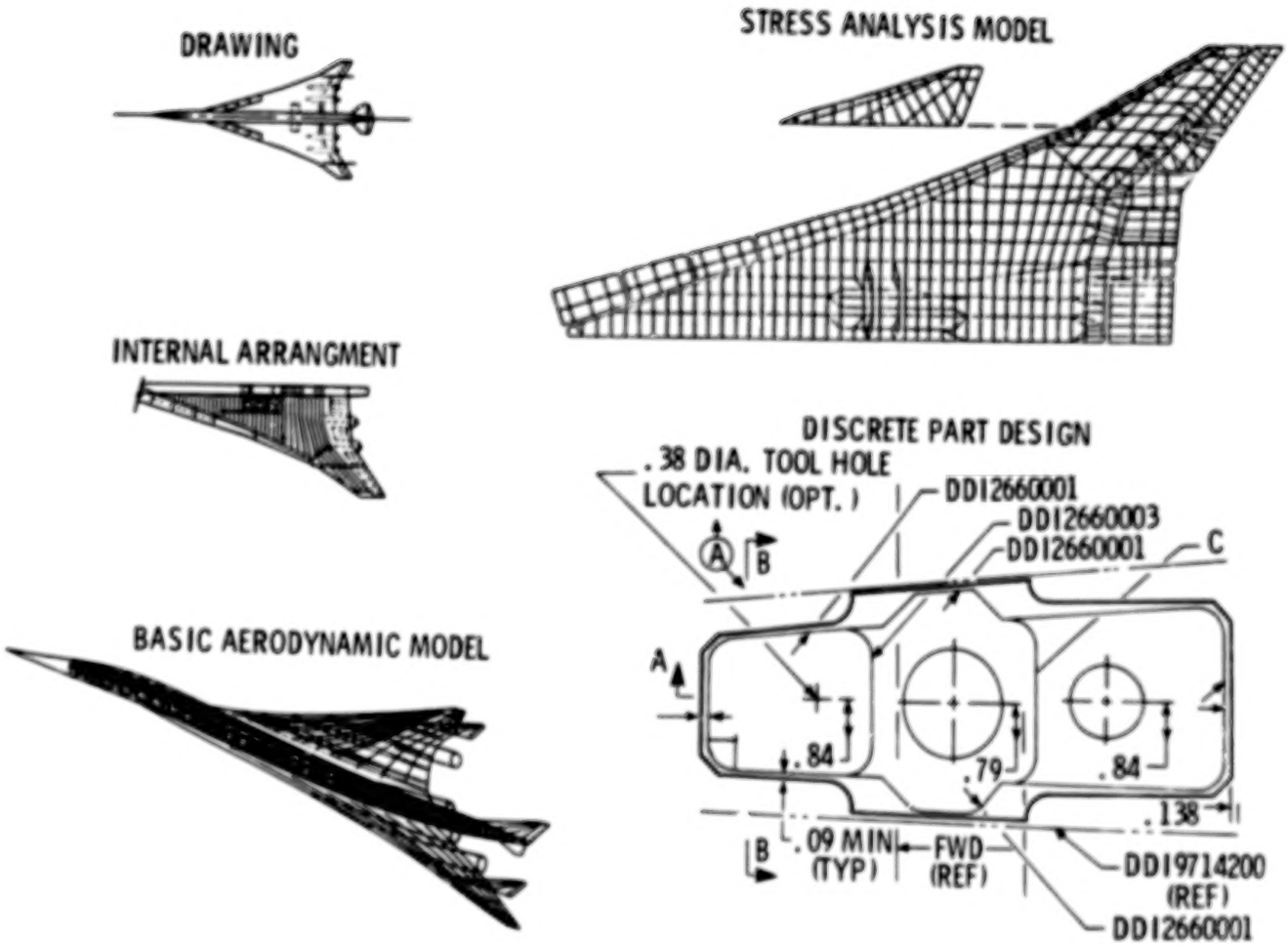


Figure 13.- Critical role of geometry information to CAD/CAM.

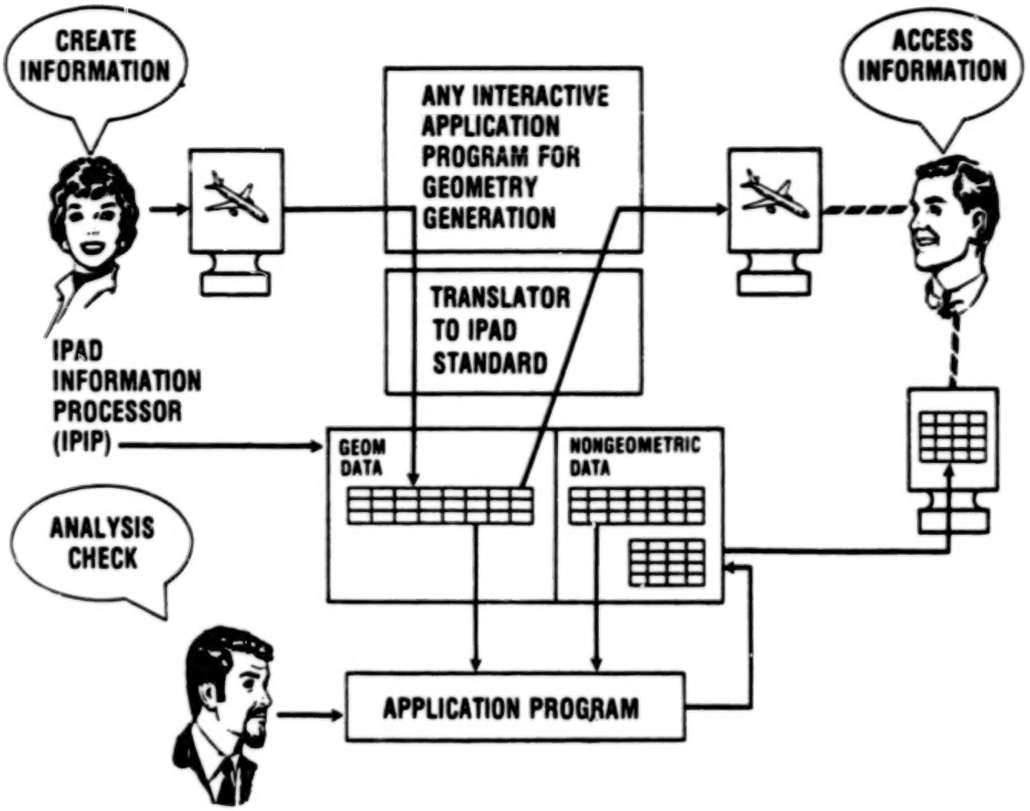


Figure 14.- IPAD approach to unified management of combined geometry and nongeometric CAD/CAM data.

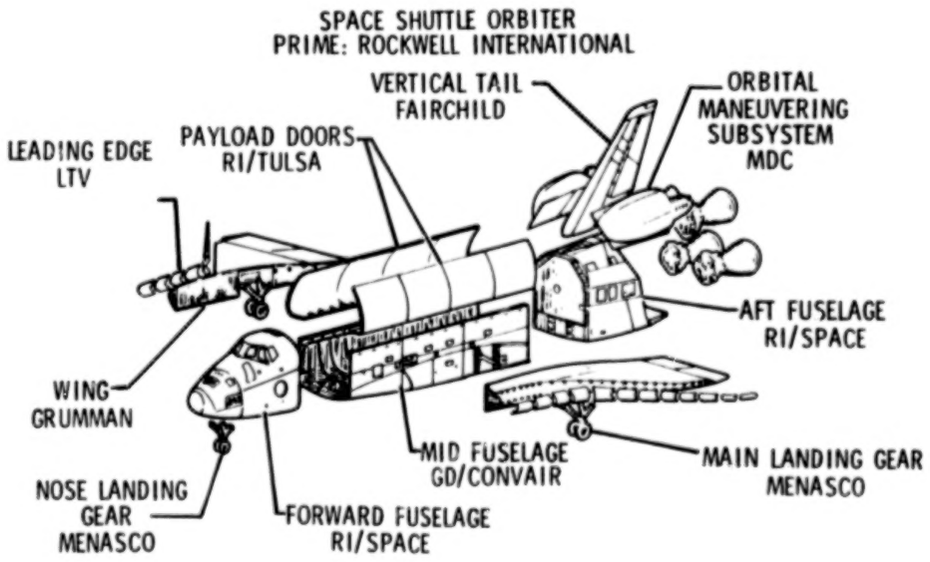


Figure 15.- Typical multicompany development approach for aerospace products.

MAJOR SUPPLIERS

RADAR	GENERAL ELECTRIC
PASSIVE DETECTION SYS.	LITTON AMECOM
COMPUTER PROGRAMMER	LITTON DSD
ROTODOME	RANDTRON
CONTROL INDICATOR GROUP	HAZELTINE
COMMUNICATION	COLLINS
IFF DETECTOR PROCESSOR	HAZELTINE
EQUIPMENT COOLING	GARRETT

EQUIPMENT LOCATIONS

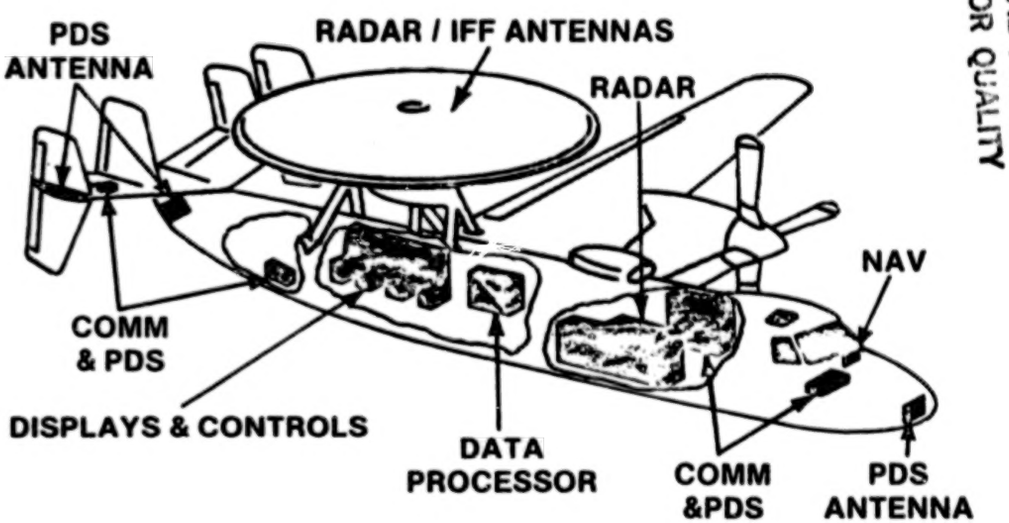


Figure 16.- Major avionics suppliers for advanced Navy aircraft E2C (courtesy Grumman Aerospace Co.).

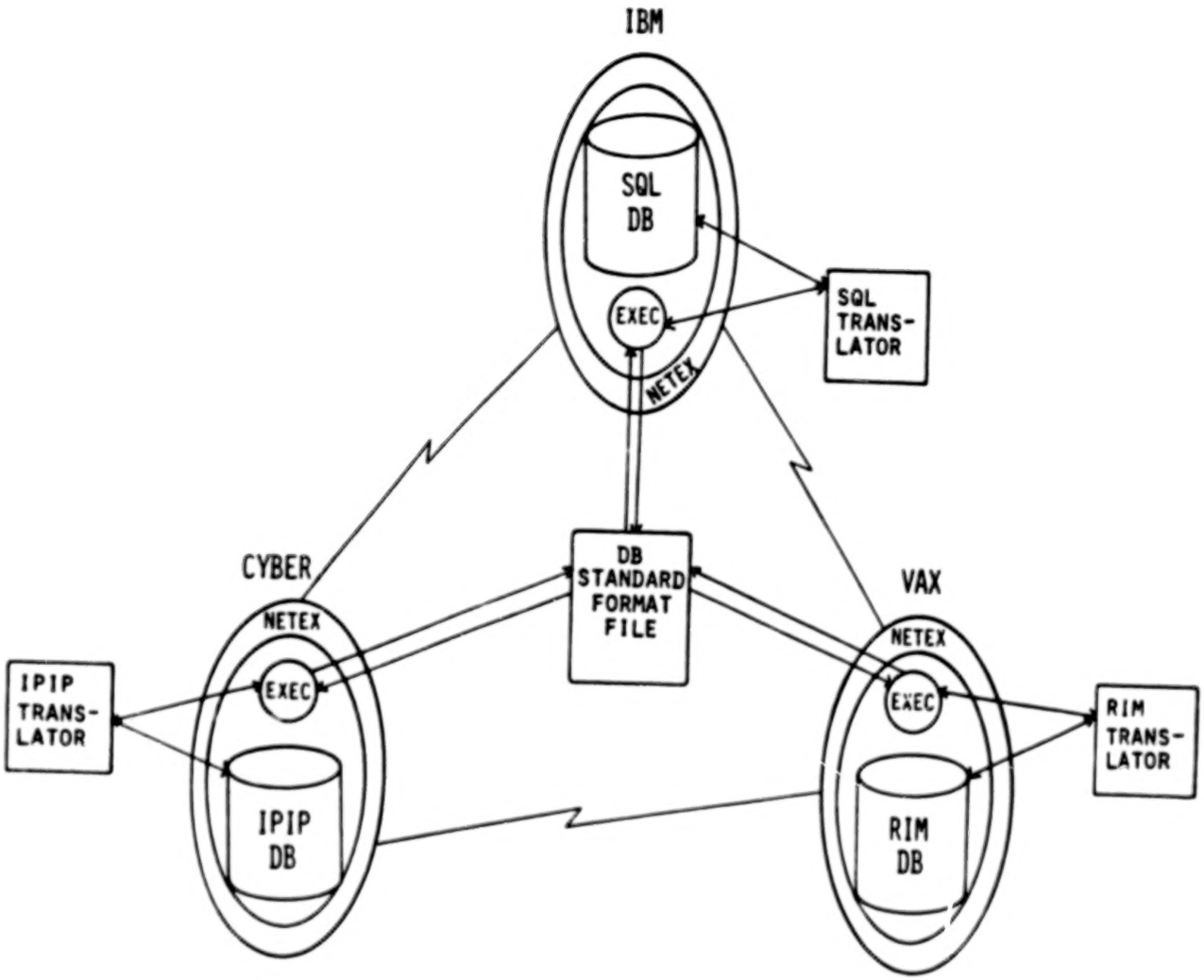


Figure 17.- Initial CAD/CAM distributed data management approach.

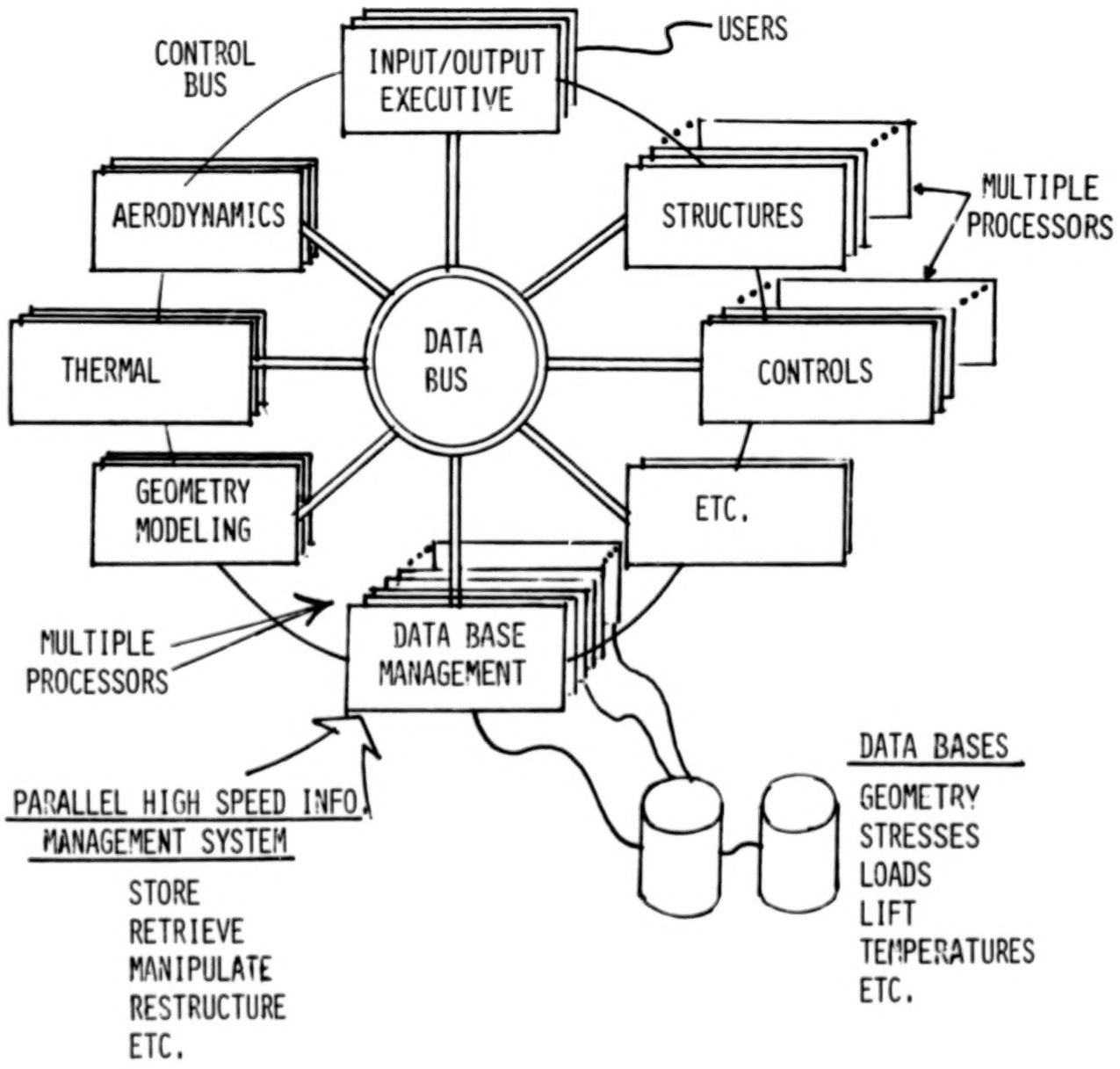


Figure 18.- Data base management in a concurrent processing multidisciplinary environment.

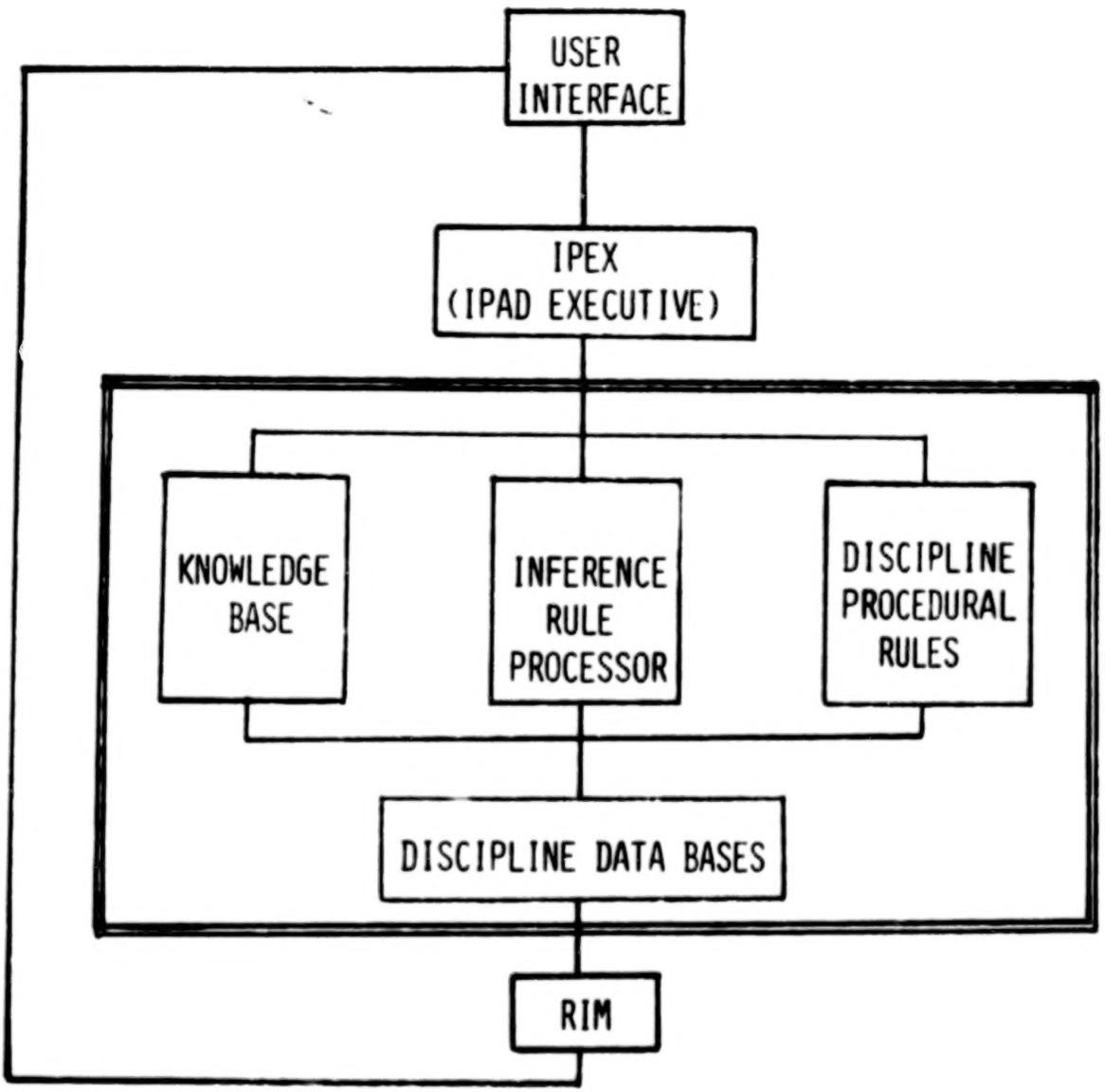


Figure 19.- Data base management approach for engineering-based expert systems.

D2

N84 22301

CIM'S BRIDGE FROM CADD TO CAM - DATA MANAGEMENT REQUIREMENTS FOR MANUFACTURING ENGINEERING

S. JEANE FORD
LOCKHEED-GEORGIA COMPANY

ABSTRACT

Manufacturing Engineering represents the crossroads of technical data management in a Computer Integrated Manufacturing (CIM) environment. Process planning, numerical control programming and tool design are the key functions which translate information from "as engineered" to "as assembled". In order to transition data from engineering to manufacturing, it is necessary to introduce a series of product interpretations which contain an interim introduction of technical parameters. The current automation of the product definition and the production process places Manufacturing Engineering in the center of CADD/CAM with the responsibility of communicating design data to the factory floor via a manufacturing model of the data. A close look at data management requirements for Manufacturing Engineering is necessary in order to establish the overall specifications for CADD output, CAM input, and CIM integration. This paper examines the functions and issues associated with the orderly evolution of computer aided engineering and manufacturing.

The concept of Computer Integrated Manufacturing (CIM) encompasses the entire spectrum of operations which define, produce, control, and support the manufacturing of a product. Computer aided systems are used in all corporate functions today; however, large scale integration of these systems is still yet to occur. Before a true "CIM" environment can exist, advances must be made in data management and database technology, hardware networking, and computerized product definition methodologies. The area which is offering an excellent opportunity for advancing the state-of-the-art in computer integrated systems is CADD/CAM (Computer Aided Design/Drafting/Computer Aided Manufacturing). Recognizing that marketing, finance, logistics, contract management, and other administrative/service type organizations play an integral role in the total "CIM" picture, this paper is focused on the technical activities which transition the product design to a producible end item. This is the function which links CADD with CAM. In most companies, the organization between design/project engineering and production is manufacturing or production engineering which covers the functions of process planning, tool design, and numerical control programming. All these activities require that the product design be transferred or integrated into another format in order to translate the "as engineered" information into an interim product definition. In essence, a manufacturing model of the engineering design must be created in the form of hard or soft tooling and work instructions.

The Manufacturing Engineering function represents the crossroads of technical data management in a computer integrated manufacturing environment. Both the CADD activities on one side and the CAM applications on the other are expanding rapidly with every advancement in computing technology. Computer aided systems for process planning, tool design, and NC programming are available. The missing element of all these systems is integration and the ability to transfer technical data from one system or discipline to another without human interpretation. This requires a data structure which allows the computer system to interpret information, make decisions, and execute application programs. Today's computer aided systems stand alone and require a human interface to input and/or translate data from one to the other. The computer has been used to make the job faster, more accurate, and less labor intensive. Automation has changed the procedures for accomplishing certain tasks, but technical information is still communicated in a traditional way. Design Engineering utilizes its computer power to conceptualize, create, and analyze the product configuration. The result of this effort in a CADD environment is a data set of geometry which is sent to a plotter which generates the traditional blueprint/engineering drawing. The computer data is sufficient to create a visual image of the design, perform some analysis of the geometry, and provide a baseline configuration envelope for tool design and NC cutter path generation. Beyond this, human interaction is still required to interpret the CADD output for technical data requirements in manufacturing. Fully automated direct CADD-to-CAM does not exist. In order to maximize the automation of

CADD/CAM, further advances in several enabling technologies must be enhanced in product definition modeling, data management, database technology, and computer hardware networking.

3-D has obvious value in analysis for design and visualization and animation for manufacturing; however, before the full benefits can be realized by CADD/CAM, a 3-D design philosophy must be established which incorporates the manufacturing product definition attributes. The output of a geometric modeling system for manufacturing must go beyond esthetic shaded images or dynamic simulation. The system must provide data. This data must be structured to provide direct input to manufacturing. A prerequisite of CADD/CAM integration is the replacement of the physical engineering drawing or electronic CADD data set with a product definition methodology that provides information which can directly feed CAM systems. This requires a geometric model which contains configuration data based on manufacturing attributes and geometry which can be associated with physical properties and other technical data/specifications.

2-D drafting systems provide little associativity of data. Current geometric modeling/3-D systems contain mathematical connectivity of geometry; however, this level of resolution in the part definition is not adequate to establish meaningful data for manufacturing. Like 2-D systems, the non-geometry data is separate. This inhibits combination of information which must be considered as a set of parameters in order to make decisions. For example, in order to select an appropriate forming

technique for a typical deep draw sheet metal part, the material gauge, alloy, temper condition, part geometry, tooling, and machine tool specifications all have to be collectively considered. This information is currently manually established from a hard copy of the engineering drawing, analyzed, and a determination made by an individual who is responsible for forwarding work instructions to the production shop. Even if a computer aided process planning system is used which is capable of selecting the forming operation, the data required to make the decision still must be established and input manually. A CADD/CAM geometric modeling system which defines a part or assembly in a manner in which an application progam can directly extract the manufacturing attributes from the model will eliminate the human interface now required.

The engineering drawing or configuration data set from a CADD system can provide basic geometry for the tool design function. However, the tooling engineer still must remodel or redefine the geometry to create tooling features internal to the part configuration and then further design the hard tooling offset from the full manufacturing model.

Numerical control programming can take advantage of the data generated in a 2-D CADD system or 3-D modeler by the project and tool designer. However, the data set or model details still must be redefined and translated into numerical control machining instructions. A part definition model which provides part feature recognition, automated

Boolean operations for machining features, surface data, tolerancing requirements, and associated non-geometric information regarding material data, process and inspection criteria is needed. With this database structure, automated NC systems can exercise machining algorithms for specific and generic features and intelligent software for ordering machining sequences, selecting cutting tools, and determining feeds and speeds.

Beyond the benefits of an integrated CADD/CAM product definition for process planning, tool design, and numerical control programming, the same data requirements can be utilized in the design and functional operation of robotics applications. 3-D modeling techniques are required to effectively establish the layout and robot/machine interface requirements. Here, graphical simulation can provide design optimization without physical mock-ups. This application also applies to the traditional assembly process.

Data management addresses the issues which are organizational rather than technical. Integrating and interfacing CADD and CAM systems and expanding to a total CIM environment requires changes in the traditional organization hierarchy. Until total CADD/CAM integration is accomplished, the Manufacturing Engineering interface between design and production will still be required. Like other areas of automation, the integration of CADD/CAM is an evolutionary process. Manufacturing Engineering will slowly move from a separate function between CADD and

CAM to a computerized representation within the design and manufacturing process. The manufacturing engineer's data requirements will be contained in the product definition generated by engineering. The knowledge of the manufacturing engineer will be maintained in expert systems which translate engineering data to manufacturing data, and artificial intelligence will be built into fabrication and assembly equipment and robotics applications to assure that the design to manufacturing cycle is carried out effectively. Manufacturing Engineering will evolve from a human activity to a computer executable function. In the meantime, it is the manufacturing engineer who must establish the specifications for CADD output and CAM input.

Database technology must define how information is identified and manipulated within the computing environment. This is a simple statement to describe a complex area of computer technology. The point which needs to be made from the technical data user community is that before data base technology can be effectively advanced to support CADD/CAM, the data requirements must be thoroughly analyzed. A typical CADD/CAM operation provides an opportunity to evaluate database considerations including geometry, version/configuration control, alphanumeric data, security, central vs distributed environments, data administration, accessibility, size, archival procedures, and many other elements which database designers must address. Additionally, the data environment must be identified with respect to the total data flow, relationship of computerized and manually processed information, optimum user interface

requirements, organizational accountability and validation procedures. The establishment of an overall functional diagram of the design/manufacturing cycle which can be decomposed to the lowest data element level is needed. Data generators and data users can be identified as well as the transfer mediums and systems for communicating technical information. The majority of CADD/CAM functions within a company and between subcontractors and vendors operate in a heterogeneous computer hardware and software environment. This is the real world. This, plus the lack of interface standards for graphics software and hardware and extensive non-standardization of product definition criteria, presents a challenge to industry, government, computer hardware and telecommunication systems vendors, and specific programs such as IPAD and ICAM. Manufacturers of computer equipment and aerospace products both want to obtain and maintain a competitive edge. The exclusive features of a superior product/system are often barriers to integration with other systems. While standardization and compliance with technical standards are logically a productive concept, they often interfere with proprietary issues, heavy investments in existing computer aided systems or market advantages. Industry, government, and vendors need to face this problem openly, with the ultimate goal the development of solutions which provide the best for all concerns. Forums such as the IPAD Program have motivated this kind of synergy. Increasing dialogue between

individual functions within a corporate structure, between the government and contractors, contractors and suppliers, and the computer industry and users will promote a closer understanding of requirements necessary for "Computer Integrated Manufacturing" to fit "the real world".

In conclusion, a close examination of the Manufacturing Engineering function can identify the elements of data management requirements necessary to achieve an integrated CADD/CAM environment. The translation of information from "as engineered" to "as assembled" has traditionally been accomplished through the interpretation of drawings, creation of work instructions and hard/soft tooling. Current CADD/CAM systems have automated established procedures but still rely on human communication of technical data. The design/manufacturing interface which occurs in Manufacturing Engineering offers an opportunity to fully exploit the benefits of Computer Integrated Manufacturing technology.

D3

N84 22302

TECHNOLOGY FOR THE PRODUCT & PROCESS DATA BASE

Robert D. Barnes
Boeing Commercial Airplane Company
P.O. BOX 3707, Mail Stop 6E-22
Seattle, WA 98124

SUMMARY

The computerized product and process data base is increasingly recognized to be the cornerstone component of an overall system aimed at the integrated automation of the industrial processes of a given company or enterprise. The technology needed to support these more effective computer integrated design and manufacturing methods, especially the concept of 3-D computer-sensible product definitions rather than engineering drawings, is not fully available and rationalized. Progress is being made, however, in bridging this technology gap with concentration on the modeling of sophisticated information and data structures, high-performance interactive user interfaces and comprehensive tools for managing the resulting computerized product definition and process data base.

INTEGRATED CAD/CAM

Conventional methods of mechanical design and production synthesize the product's definition and its realization as a certified manufactured entity through the use of abstractly represented information, e.g., engineering drawings, pictures, components references, lists of discrete data items and analysis results. These "paper" methods, though enhanced by the application of computerized tools for information creation and management, have practical limits of effectiveness.

In the present environment where increased automation is one of the few - if not the principal - avenues to higher productivity, expanded computerization of conventional methods alone will not provide enough leverage to counter adverse trends in product cost and quality.

Certain departures from present engineering procedures, attended by the introduction of supporting technological capabilities, are required to reach the levels of automation at which significant increase in productivity is achieved. The industrial approach which embodies the prescribed procedures and technology is most often referred to as integrated CAD/CAM. The use of special purpose computing systems to assist the functions of design and manufacturing engineering is not novel here, nor the distinguishing issue per se. The key aspect

of the approach is the integration of such computing functions in an overall process that takes on the character of a single system. This integrated system should operate at the enterprise level with a configuration managed digital product definition and process data base. The most important features of the integrated CAD/CAM system are: (1) its structured approach, (2) its unified processes, (3) concrete (realistic) data models, and (4) normalized information structures.

Structured Approach

An integrated CAD/CAM system, like any other system, exists as the sum of its parts. Its internal description is decomposed into separate functional pieces which are, in fact, individual computing applications supporting specific design and manufacturing tasks of classical categories. These organizational or task-specific functions have to a large degree been treated as self-contained and have in many cases been well served by individual CAD or CAM subsystems. Integrated CAD/CAM, however, emphasizes the logical relationships among function-specific subsystems in the context of an overall industrial process. This is especially true with regard to the (basically one-way) flow of data through the product development cycle and the position of an individual application relative to upstream and downstream subsystems and users. There is, then, an ordered structure to integrated CAD/CAM that corresponds basically to the structure of a given product development process, and in which individual CAD and CAM applications are fashioned as logical parts of the whole, in contrast to having a strictly stand-alone role.

The interdependencies of integrated CAD/CAM subsystems are readily perceived in examining the informational and functional interface between engineering and manufacturing. Most students of the problem have correctly concluded that a CAD/CAM interface at the data level only (i.e. the passing of design information in some agreed-to computer sensible format from engineering systems to manufacturing systems) is an inadequate modicum of "system integration." The technical and operational issues go considerably beyond the transfer of data, and include, among others, configuration control of the engineering design model, design change tracking, the direct utility to manufacturing of the information structure and semantics of the data being transferred (is the design intent systematically interpretable?), and engineering and manufacturing systems compatibility with respect to representation, manipulation and evaluation of the design data model.

An engineering systems to manufacturing systems interface at the data level, however relatively productive, does not, a priori, constitute integrated CAD/CAM. The technical and operational factors involved are systemic. The features and characteristics necessary in subsystems in order for the total

system to sustain an integrated process must be designed into those subsystems. The technological catalysts of integrated CAD/CAM - unified processes, concrete data models and normalized information structures - cannot, in most cases, be added on to, or salted into, a given repertoire of CAD or CAM subsystems. This means, generally, that integrated CAD/CAM for a given industrial process must be approached from the top down in the sense that certain in-use subsystems must be replaced (redesigned, rebuilt) with others which embody the required technological common denominators and inter-system operational relationships. The integrated system must be designed and built from scratch, not synthesized from a potpourri of data-linked existing components.

The exampled system interdependencies between engineering and manufacturing occur also within each of these two disciplines. The same integration considerations apply with respect to subsystems supporting the engineering functions of product conceptualization, product design, product analysis and product testing, as well as the manufacturing subsystems for process planning, tooling, process control and quality assurance.

Unified Processes

Integrated CAD/CAM is fostered when the technical processes involved are unified. The requirement for the unification of processing functions derives from the integration concept of shared common data, wherein individual CAD/CAM applications operate with an evolving product definition represented in computer-sensible form that is standard with regard to information content and data format. Ideally, all end-user computing applications in an integrated CAD/CAM environment either use the common data model directly in performing their functions or transform it into the required representation as a secondary form (e.g., a finite element structural analysis model). Transformations to a nonstandard form are limited to those applications that require alternate representations for functional purposes, and in these cases, the system should effect an automatic transformation, if practical, in avoidance of manual interpretation and reconstitution of the data model.

The direct use of common data by many different applications leads to a normalization of the common utilitarian processes found in these applications. A unified set of system functions to create, manipulate, evaluate, store and access common standard data is easily rationalized in an integrated CAD/CAM environment. The benefits of unification are substantial, not the least of which is the reliability and consistency achieved through concentrating available resources on the support of a comparatively small set of system functions.

Concrete Data Models

A principal objective of integrated CAD/CAM in the industrial process is to replace abstract information, which requires human interpretation for precise meaning, with concrete information which much more closely represents physical reality and which is informationally rich enough to sustain algorithmic evaluation. This transition in information structure is one of change from two-dimensional representation of product geometry to three-dimensional forms thereof or, more to the point, geometrically complete representations instead of incomplete ones. The most obvious example of two-dimensional representation, and by far the most pervasive medium of product description, is the orthographically projected engineering drawing. This information medium, if complete and unambiguously rendered, when accompanied by any required reference or discrete digital data and when contextually treated with regard to spatial orientation, permits precise human interpretation and serves as an adequate form of product definition when so interpreted. The engineering drawing, however, even in computerized digital form, is virtually worthless as a mathematical model of physical objects in the context of using such models to automate engineering and manufacturing functions. The 3-D edge-vertex ("wire frame") models common in today's generation of interactive graphics CAD systems, though more useful than 2-D drawing models, also impose definite technological limits in attempting to move to more sophisticated integrated CAD/CAM applications. A geometrically complete three-dimensional description is required for such purposes, to the end that a designed product's spatial, topological, volumetric and mass properties are intrinsic to its computer-sensible representation and can be evaluated directly. Examples of such methods include "solid geometric modeling" wherein volumetric building block primitives are combined through Boolean operations to represent homogeneous "solid" objects of arbitrary shape or physically complete 3-D objects are represented by manifolds of contiguous analytic surfaces which form their shape and enclose their volume.

Integrated CAD/CAM is based on a computer-sensible three-dimensional product definition data base. This concrete information structure is targeted to replace the engineering drawing and other abstract data as the master source of engineering design and manufacturing information. Geometrically complete product definition is the key to higher orders of industrial automation. Such technology will permit the employment of scientific methods to a markedly higher degree than is now the case.

Normalized Information Structures

As industrial and business computing applications have expanded and matured over the years, the nature of the data stored, manipulated and processed by these applications has undergone a metamorphosis. Early computing applications were dedicated almost exclusively to (1) scientific cal-

culations or (2) the management of data. The latter, in its beginnings, was not much more than a replacement of manual methods with computerized means for the filing and keeping of records. Over time, however, applications using these data have reached certain levels of processing "intelligence" with regard to the data's content and meaning - a built-in system cognizance of the informational aspects of the data. When individual data elements or collections thereof are treated as information for given purposes, a higher-order structure is ascribed to the compendium of data being so treated. These application-specific "views" of a collection of individual data elements or lower level data structures are termed information structures.

Industrial production control computing applications (e.g., parts ordering, material requirements planning, production scheduling, process and operations planning and control, inventory management, etc.) are examples of more sophisticated and mature computerized functions in which the pertinent information structures are fairly well understood. The data conventions, relationships, definitions and semantic constructs for these types of applications have achieved a de facto degree of standardization and normalization (a common and basically non-redundant completeness), at least for a given company's purposes and, generally, for individual industrial categories as a whole. The informational value of data items like "work center," "lot size" or "production line capacity" is real and immediate to applications which use the normalized conventional meaning of these items in logical processes. This exampled environment contrasts sharply with that of today's operational CAD/CAM systems in which we are still struggling with the ramifications of primitive data structures and, with few exceptions, have not addressed at all the next level of sophistication and leverage --- information structures.

Information structures can be looked at as access paths into a product definition data base or, conversely, as a hierarchy of decreasingly abstract informational constructs with the most abstract being at the top of the tree. A simple example of such an information structure cum access path follows:

```
AIRPLANE
 WING
  WING BOX (central structural section)
   WING BOX RIB NO. 4
    RIB WEB
     WEB RIVET PATTERN
      RIVET HOLE NO. 42
       x,y,z COORDINATES (of RIVET HOLE)
```

The structure indicated is obviously a part of the taxonomical breakdown of a specific airplane design. Each branch in this taxonomy is a cache of ever more

detailed information. The RIB WEB, for example, is a mono-detail sheetmetal component which has a generally standard shape and topological characteristics: an airfoil-like profile, various cutouts and rivet holes for attaching other components. The RIB WEB itself, then, has a (parametrically at least) definable and normalizable information structure, one feature of which is shown in the example (WEB RIVET PATTERN). The other information levels in the exampled structure may be treated similarly, that is, decomposed into informational features with ascribed conventional meaning and content. The advanced integrated CAD/CAM applications needed to move productivity forward require data models with intrinsic informational value. The importance of this factor cannot be overestimated. Unfortunately, we have made little progress along these lines in the CAD/CAM world, especially in the mechanical design and fabrication disciplines. We deal mainly with primitive data structures - points, curves, surfaces, dimensions, text, etc. - elements which of themselves have no intrinsic meaning or informational value.

There is, of course, no common CAD/CAM information structure which permeates industry, mechanical products or otherwise. Information structures differ among industrial product categories, indeed even among different companies producing the same industrial products. Each CAD/CAM facility must develop and rationalize its somewhat unique information structure. The process amounts to capturing in a systematic way what already exists, not inventing something new or conforming to an illusive norm. Also, information structuring has nothing directly to do with computer science or computing systems development. It is basically a plain old garden variety industrial engineering job. One, however, which must be performed before a more effective integrated CAD/CAM computing enviornment can be developed and implemented.

CONCLUSION

The basic technology needed to support the more effective methods of integrated CAD/CAM, especially the concept of a "computerized master model in a shared data base context," is not fully available and rationalized. Significant efforts to bridge this technology gap are underway in The Boeing Company, as well as in many commercial, academic and governmental institutions elsewhere. These efforts tend to concentrate on four fundamental technological areas deemed to be critical in moving industrial automation forward through integrated CAD/CAM:

1. Data management systems of significantly more capability than presently available to support large-scale scientific and business oriented data bases, in a distributed mode if necessary. These DBMS products must provide a

high degree of application independence and must support multiple user and application data schemas and relational, hierarchial, and network type data structures.

2. Geometrically complete product definition information and data structures and mathematics.

3. Unified CAD/CAM data modeler - a "geometry engine" to create, manipulate, view and evaluate the 3-D product definition data base.

4. Intelligent distributed processing communications network - a delivery vehicle to bring integrated CAD/CAM to the end-users.

MATERIALS PROPERTIES DATA BASE COMPUTERIZATION

R. G. Baur, M. L. Donthnier,
M. C. Moran, I. Mortman, and R. S. Pinter
General Electric Co.

SUMMARY

Material property data plays a key role in the design of jet engine components. Consistency, accuracy and efficient use of material property data is of prime importance to the engineering community.

This paper describes the system conception, development, implementation, and future plans for computer software that captures the Material Properties Handbook into a scientific data base. It allows the engineering community to access raw data and property curves, display multiple curves for material evaluation and selection, allow direct access by design analysis computer programs, display the material specification, and maintain a historical repository for the material evolution. The impact of this activity includes significant productivity gains and cost reductions; all users have access to the same information and provides consistent, rapid response to the needs of the engineering community.

Future plans include incorporating the materials properties data base into a network environment to access information from other data bases and download information to engineering work stations.

INTRODUCTION

The seven volumes of the Material Properties Handbook (or, as it is often called, the "Redbook") constitute an encyclopedia of sorts -- but one that is used only by AEBG employees. What these volumes contain is a complete set of material properties covering all the materials used in GE aircraft engines. The properties of each material are presented for use in the design, analysis, and manufacture of GE's aircraft gas turbine engines and engine parts. The volumes of documentation include 3500 curves, a tremendous amount of detailed data. And each Redbook curve consists of a line (linear or curved) or a set of lines, including average, average minus three standard deviations (3[sigma]), and average minus k standard deviations (95/99), with descriptive information defining a given population. Obviously, this represents a considerable challenge in terms of keeping the records accurate and up to date.

For years the seven volumes of material properties that officially comprise the Redbook have been maintained manually. This has been a time-consuming and rather costly process, involving retyping and republishing whenever new materials, processes or specifications were introduced. The costs of printing, binding, and distributing the handbooks add up rapidly as they are reissued time and again. There are difficulties in controlling copies so that designers and analysts use the right data. But until recently there has been no other way to maintain the materials records.

Now the situation has changed: computer-aided engineering technology has been used to eliminate the need for a labor-intensive, manual system, substituting

in its place an on-line, real-time engineering data base. Using the computer to store and present the information currently housed in the seven-volume materials handbook will not only be cost effective, it will broaden the usefulness of the materials system as a whole.

This paper is a status report on the ongoing project to incorporate state-of-the-art CAE technology into the latest revision of the Redbook. The system should be partially available by December of 1983 and fully operational in 1984.

STRATEGY

The basic plan of attack consists of the following steps:

1. Establish a scientific data base.
2. Enter the curves with their descriptive information into the data base.
3. Develop user-oriented application programs.

THE DATA BASE - The FOCUS data base was chosen for the project. FOCUS's main attributes are that it is flexible, user friendly, and compatible with the Engineering Materials Technology Laboratories' other data bases. In addition, it can be enlarged easily and can provide the user with rapid access to the data it holds.

CURVES - Before they are entered into the data base, the material property curves presented as average and -3[sigma] are restructured to the new statistical format of 95% confidence of 99% exceedence (95/99). This is accomplished by digitizing the existing curves via the Computervision Interactive Graphics hardware.

USER-ORIENTED APPLICATIONS - Discussions were held with the user community to define present and future uses of the materials properties. From the comments of a wide range of users, various system requirements were defined.

For one thing, the system software will be structured to make inquiries as easy as possible. Two modes of entry will be provided: If the user knows his material property curve number, he can use it to obtain direct access to the vital information. If he does not know the curve number, he can resort to a tutorial mode of entry. This will allow users to work their way sequentially through the table of contents to the curve number they want. The table of contents includes alloy designation, material property, and curve number. Users will be able to employ any of these descriptive factors to locate the information they need.

Users will be able to repeat the process until they have established the curve numbers they want to investigate, up to a maximum of ten. Since the system functions in a time-sharing mode, access to the data is quick and cost effective.

The output selection menu, shown in Figure 1, will appear on the CRT screen on command. The menu will list various types of information from which the user can choose. The requested information will then appear on the screen and will also be available as hard copy output from a hard copy unit.

The first category of information is a display of the average, -3[sigma] (i.e., "minimum"), and 95/99% confidence lines for a material property. Such a plot is shown in Figure 2, which is a reproduction of the curve as it appears in

the hard copy manuscript currently in use.

Figure 3 is a table of the data points that make up the three lines shown in Figure 2. The user can see these data points simply by requesting them from the output menu. The average and -3[sigma] points are digitized values, while the 95/99% confidence points result from the algorithm processed against the average points.

Figure 4 displays the average, -3[sigma], and 95/99% lines with the raw data points superimposed on the average line. The raw data points are a product of the materials testing process and are available on a "need only" basis.

One of the factors most likely to fluctuate is a material's cost per pound. As a result, the purchasing data base will be accessed regularly and the up-to-date cost figures stored in the system. By requesting current cost data from the system, the user will receive the latest available figures on the screen or, if desired, in the form of hardcopy printout.

One of the most useful features of the system will be the user's ability to make rapid and accurate comparisons among multiple curves. Figure 5 shows how two curves with average lines would be displayed. (The system can handle up to 10 curves at a time.) For making such comparisons, an automatic curve-reading feature has been incorporated into the software. Given a particular temperature, the software picks off the appropriate stress levels for the curves being displayed and prints the results.

Information searches will be a user-controlled option. The user might ask the system, "What materials have a .2% tensile stress greater than XXXXX MPa at a temperature of XXXX^{o}C?" The system would then automatically search the entire materials data base to retrieve all materials meeting such conditions. These would be printed out on the screen.

FUTURE PLANS

Future plans include incorporating the materials properties data base into a network environment to access information from other data bases.

Utilizing an engineering work station, a design engineer can create and analyze a design using solid modeling - finite element analysis technology and properties obtained from the materials properties data base. The results of the design and analysis are stored in the analysis data base.

Since FOCUS is available on IBM PC's, portions of the main materials data base can be downloaded to a PC for local analysis. This concept will improve computer response time.

BENEFITS

UNIFORMITY - This system will allow authorized users throughout the engineering and manufacturing community to use the same data. It will embrace the major AEBG sites of Lynn and Evendale plus all of the various satellite plants and repair shops. As a result, there will be an unprecedented level of uniformity within AEBG for all materials-related data.

AVAILABILITY - In addition, the data will be more available than ever before. By doing away with seven cumbersome volumes, the new system will put materials information in the hands of any authorized user with access to a CRT terminal or a remote printer. Ultimately, the result will be a paperless system with computer-developed reports as a by-product. The major computer programs in stress, vibration, and heat transfer analysis will communicate directly to obtain the material properties. This not only guarantees the availability of this information to the analysts, it also assures that they will be able to run their programs with the most detailed and accurate materials information available.

PRODUCTIVITY - Finally, the new computerized system will result in major productivity improvements and in reduced costs, both from the user's standpoint and from the standpoint of the organization responsible for analyzing the materials data, generating the curves, and publishing the results. By improving the way material properties are stored, retrieved, and presented, the computerized system will transform a labor-intensive system into a state-of-the-art data base for all the needs of engineering and manufacturing.

```
**************************************************

                    MAIN MENU

 1.  PLOT AVG, -3S & 95/99 LINES FOR ONE CURVE #
 2.  TABLE OF DATA POINTS
 3.  READ CURVES -- POINT PAIRS
 4.  MATERIAL COST PER POUND
 5.  OVERLAY PLOTS
 6.  PLOT OFFICIAL REDBOOK CURVE
 7.  CHANGE CURRENT KEYSTRING (CURVE NUMBER)
 8.  PRINT ALL KEYSTRINGS (CURVE NOS.) DEFINED
 9.  TERMINATE REDBOOK DATA FUNCTION

**************************************************
```

Figure 1.- Output selection menu.

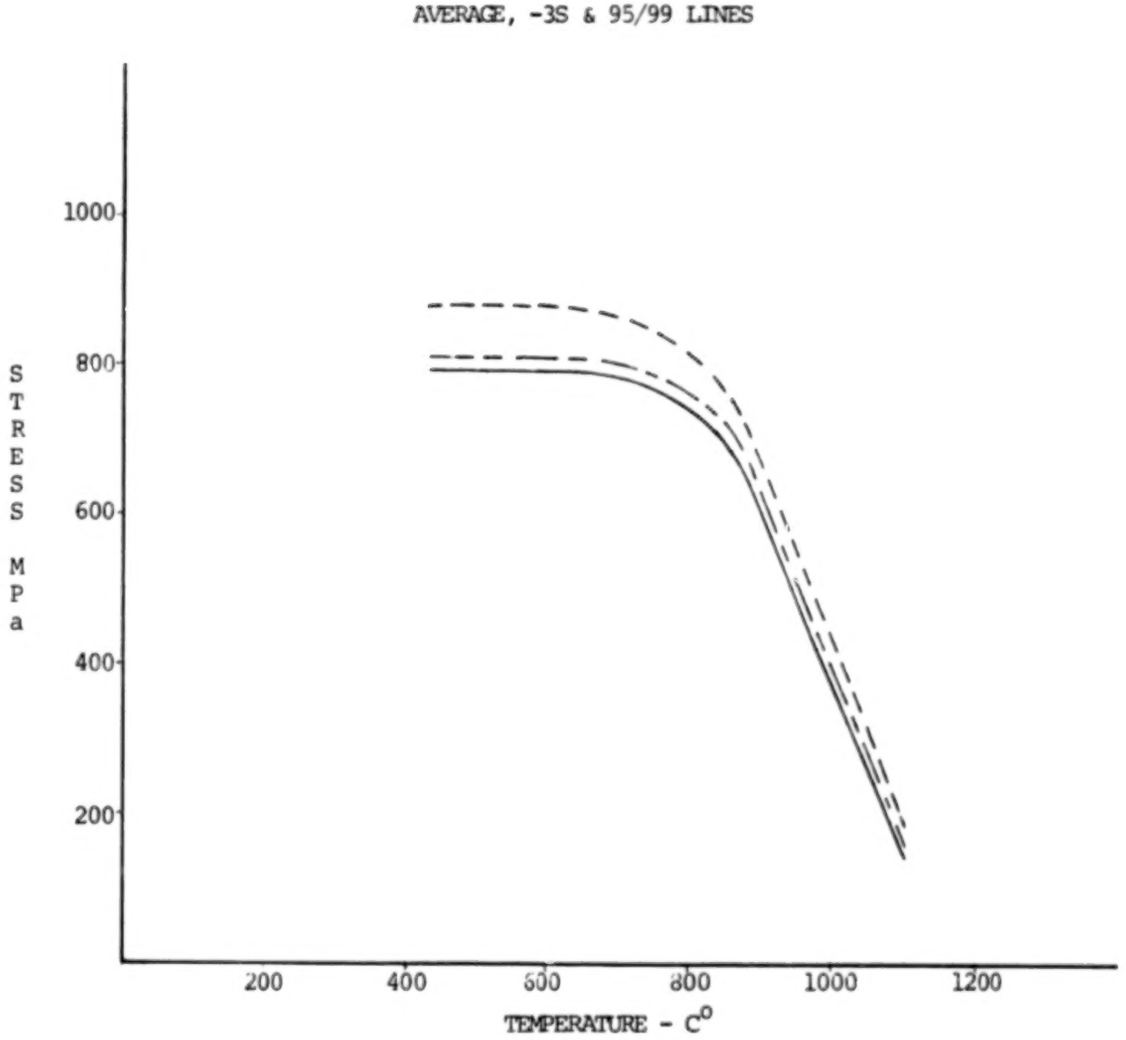


Figure 2.- Output display.

AVERAGE		MINUS 3S		95-99	
TEMPERATURE - C^{o}	STRESS - MPa	TEMPERATURE - C^{o}	STRESS - MPa	TEMPERATURE - C^{o}	STRESS - MPa
430.32	874.91	430.32	806.23	430.32	790.17
504.12	874.28	504.12	806.37	504.12	790.51
577.91	876.56	577.91	805.68	577.91	789.13
651.72	874.91	651.72	803.06	651.72	786.31
725.51	860.08	725.51	793.20	725.51	777.62
799.31	819.68	799.31	761.41	799.31	747.83
873.10	729.35	873.10	681.71	873.10	662.13
946.90	564.91	946.90	514.98	946.90	503.33
1020.70	377.85	1020.70	330.41	1020.70	319.31
1094.49	195.33	1094.49	151.75	1094.49	141.71

Figure 3.- Data points.

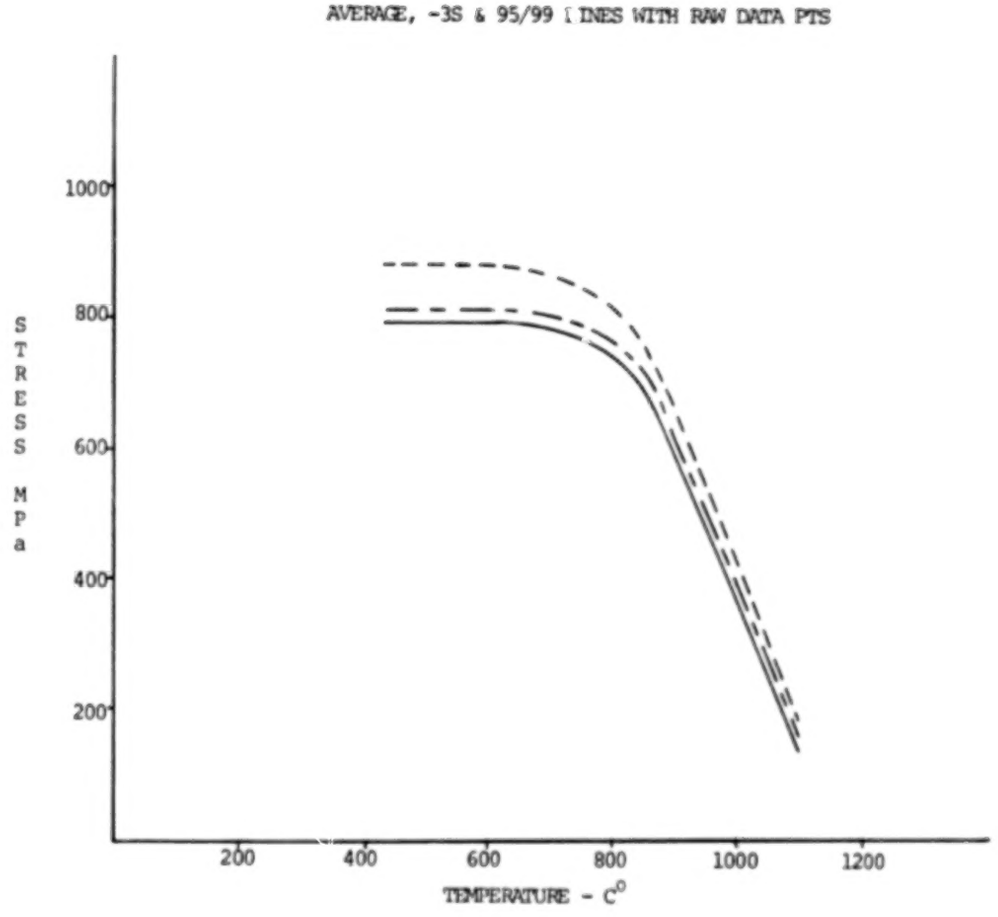


Figure 4.- Output display with raw data points.

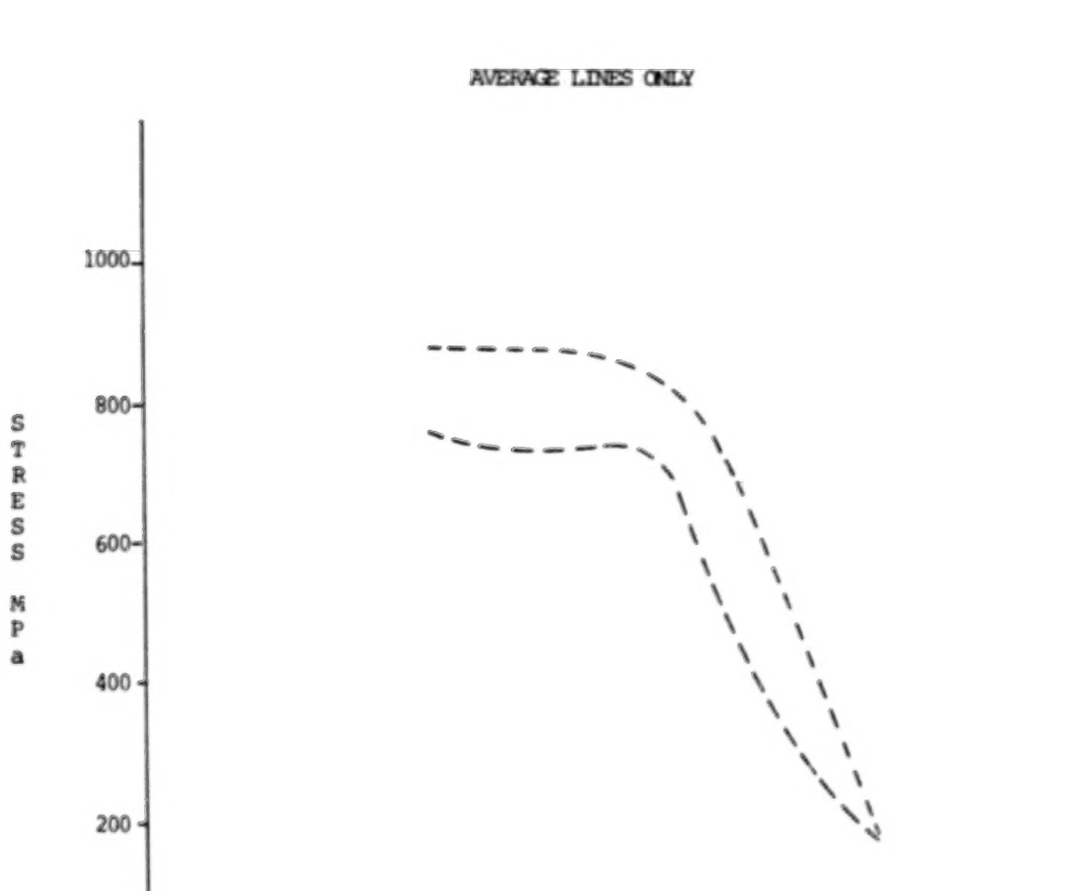


Figure 5.- Comparison of two materials.

D5

N84 22304

DESIGN VERSUS MANUFACTURING DATA BASE MANAGEMENT REQUIREMENTS

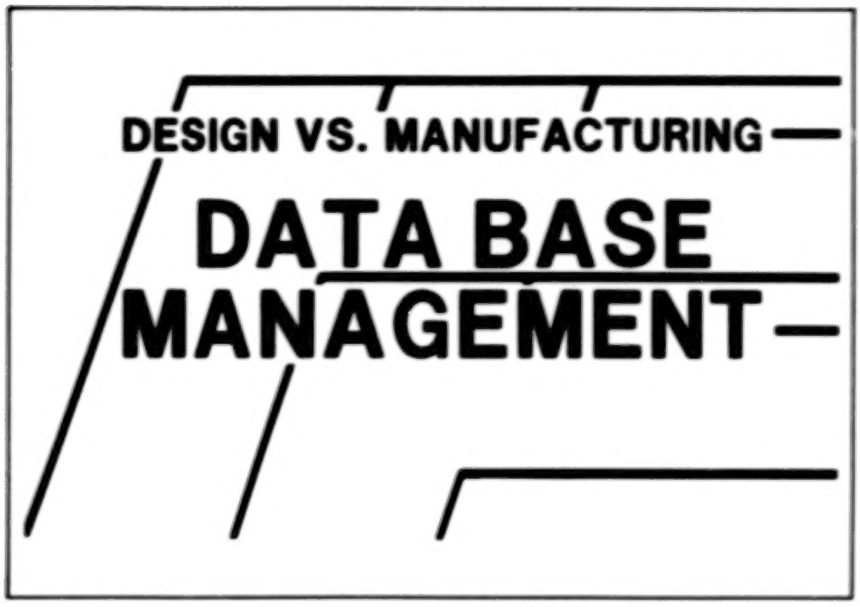
E. G. McKenna
BOEING COMMERCIAL AIRPLANE COMPANY

ABSTRACT

Data base management systems have proved to be valuable manufacturing and design tools as these disciplines are exceptionally information intensive, requiring precise organization and control of data processing and utilization. One such data base manager is the IPAD* system, which was originally developed to support the design process but has been expanded to incorporate the additional needs of manufacturing. To set the stage, an overview of the design and manufacturing process is presented. The different functions of computers in these processes are then discussed. Finally, the design and manufacturing requirements for a data base manager are compared and contrasted.

* IPAD (Integrated Programs for Aerospace-Vehicle Design (development is performed by The Boeing Company under NASA Contract NAS1-17555. IPAD software and documentation may be obtained from the IPAD Program Management Office, The Boeing Company, P.O. Box 24346, Seattle, WA 98124, M/S 73-03.

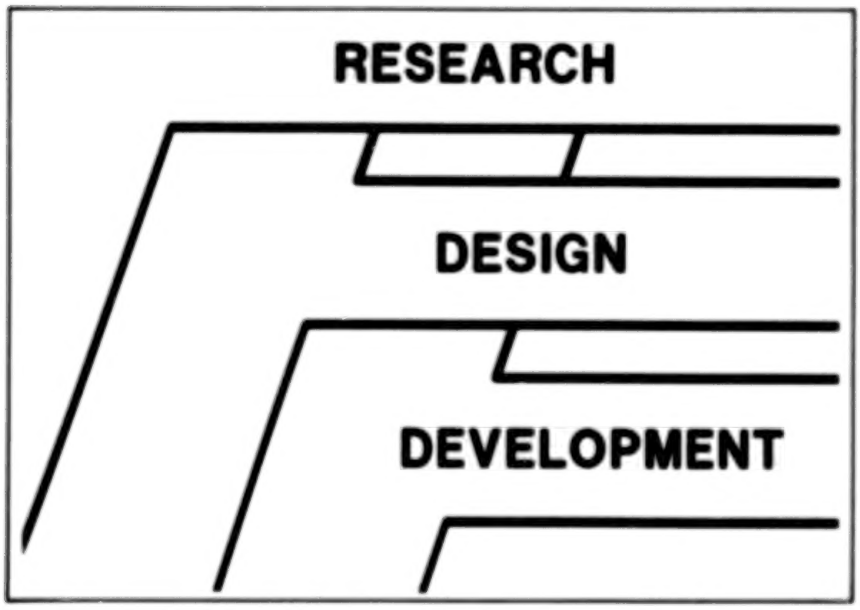
PRECEDING PAGE BLANK NOT FILMED



INTRODUCTION

As originally envisioned, IPAD was designed to support the data base management needs of the design process, but the project charter has gradually expanded to include manufacturing as well (refs. 1, 2, 3). This presentation will take account of the similarities and differences inherent in the design and manufacturing processes and the requirements of the data base manager needed to support them.

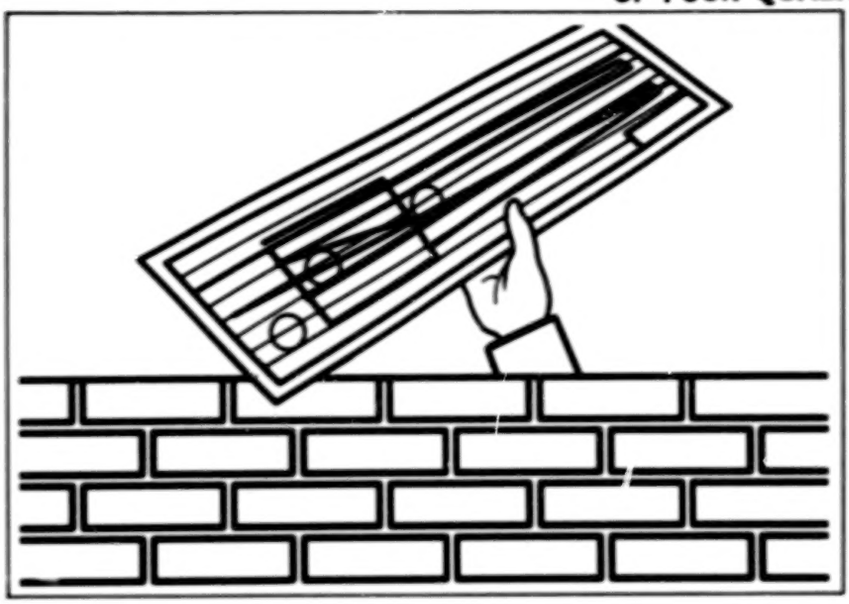
Let us be clear about what is meant by design. As used here, design is a desired arrangement of items and activities needed to satisfy a specific function. The design process, then, consists of methodologies and procedures used to create a design. The IPAD staff has documented a design process description that was used as a guide in constructing the IPAD information processor (IPIP). This description should not be construed as the only way to view the design process, but it does provide a framework upon which IPAD software can be built.



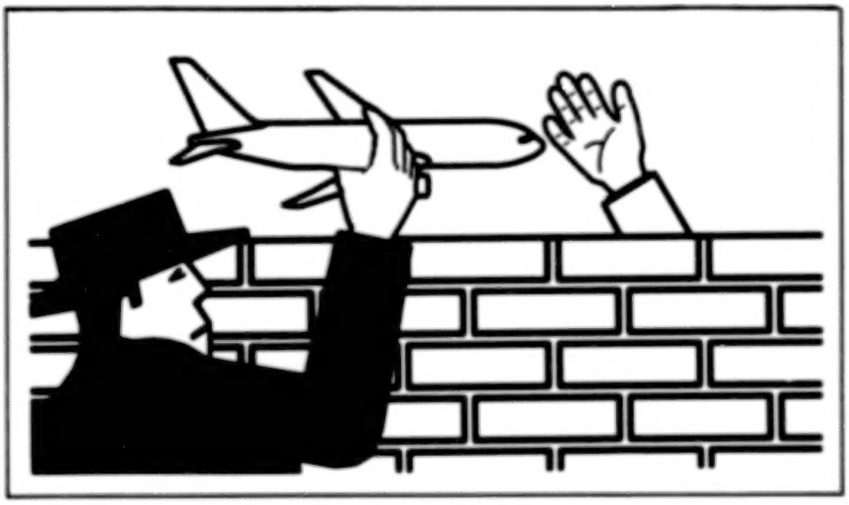
In the aerospace environment the design process can be thought of as progressing through three phases. Phase 1, continuing research, embraces long-term research activities that continually develop new design procedures, technical analysis capabilities, material properties, and other technological advances. Phase 2, preliminary design, consists of the initial design efforts in which design criteria are selected and designs are sized, refined, and verified. Phase 3, product development, is the stage during which detail design is completed and the product is manufactured, verified, and supported.

These three phases are not simply a linear progression, for there is a great deal of overlap and the activities of the three phases often occur in parallel, as schedules are very tight in aerospace and design activities cannot wait for the completion of a prior task. Each activity within the design phases can be subdivided into preliminary, detail, and verification design.

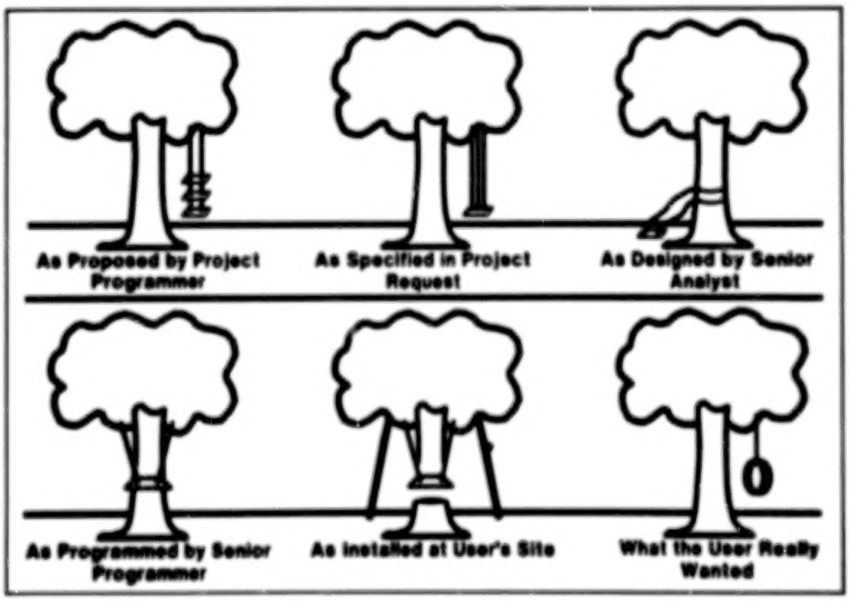
It is interesting to note that in phase 3, the product is actually manufactured, which suggests that manufacturing is part of the design process.



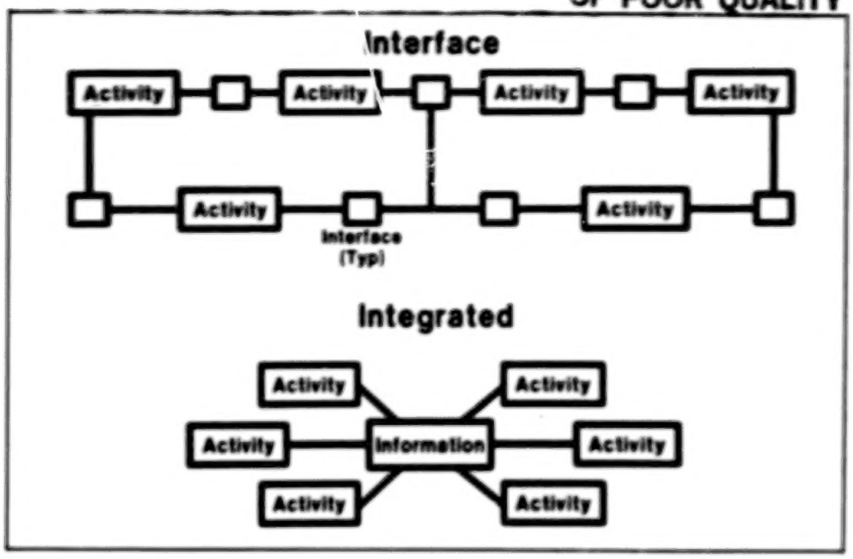
...Historically, manufacturing has been viewed as an activity that must be supported after the real design work has been done (ref. 4). Design's attitude used to be that the drawings, when completed, could be "thrown over the wall" to Manufacturing, whose job was to build the product.



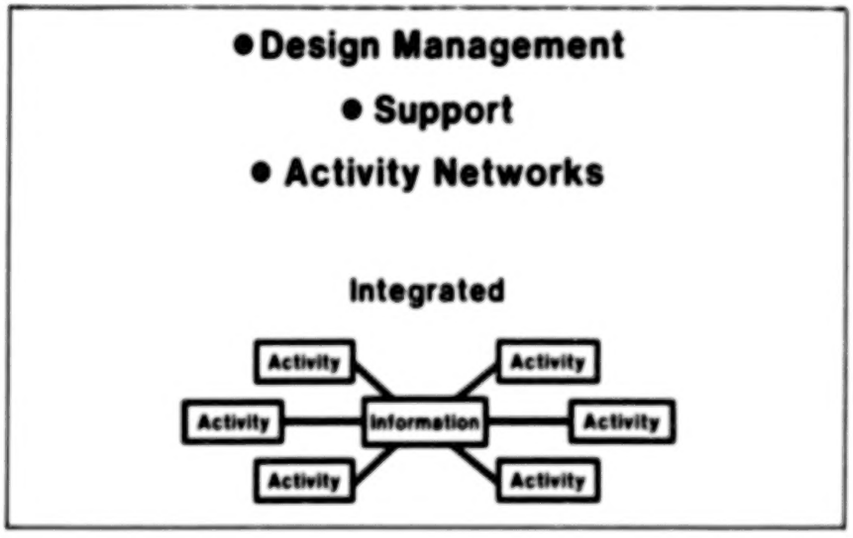
...Design then served as liaison to resolve discrepancies and correct design errors discovered during production.



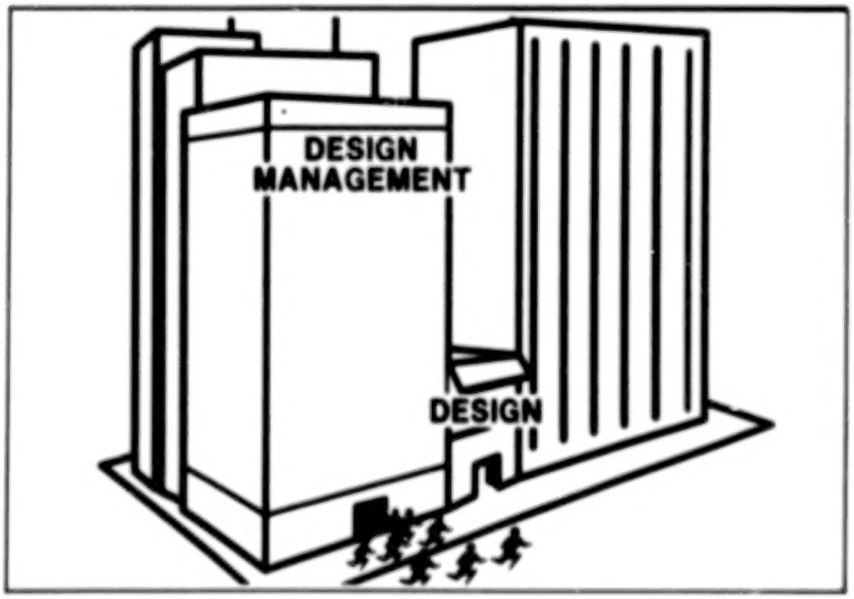
...This vague interaction is rapidly giving way to an integrated process.



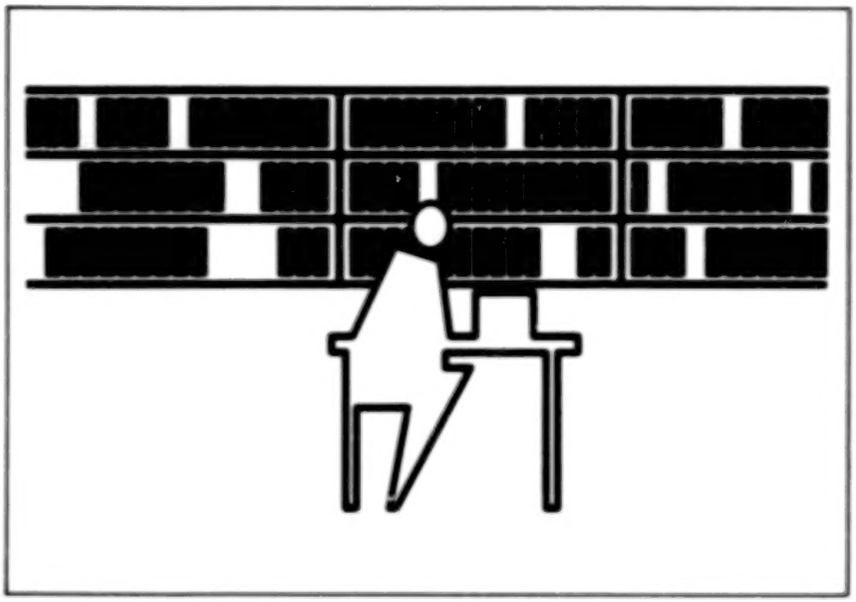
Let me distinguish here between the terms "integrated" and "interfaced." By "interfaced" we mean that information is transferable between activities or machines, with the implicit requirement that the information must be reinterpreted or reformatted from copies of the original. "Integrated" means that information is shared by activities or machines, which requires that it be managed so that its integrity is ensured by the automatic control and distribution of updates.



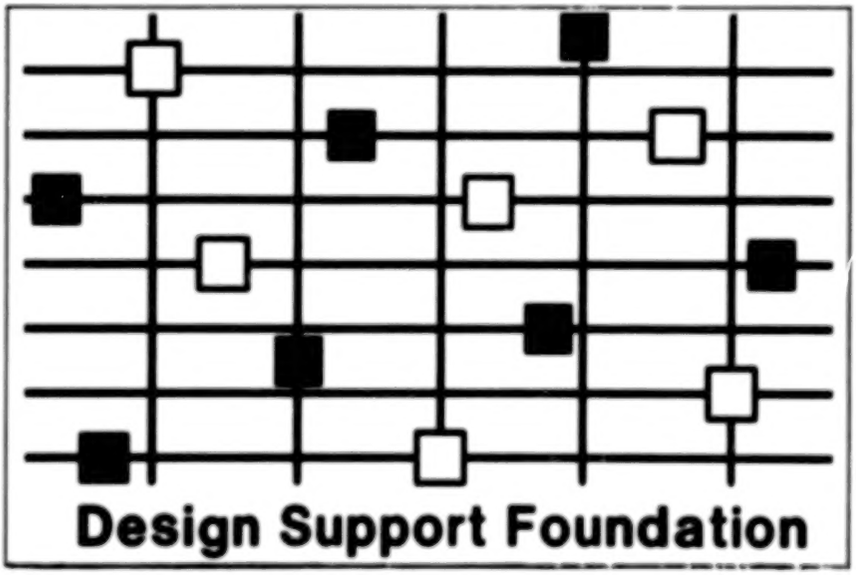
The design process framework described herein is controlled and stabilized by a foundation consisting of design management, support, and activity networks.



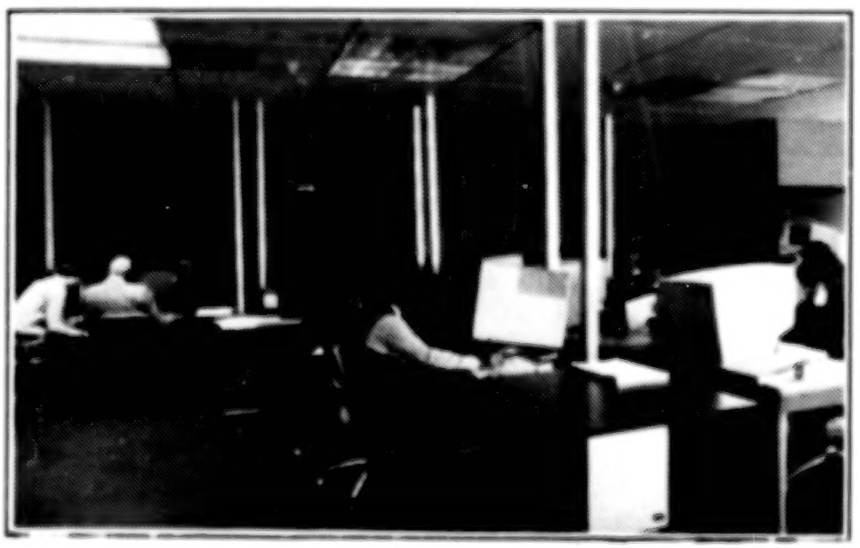
Design management deals with the management plans and controls, operating procedures, and reports necessary to support the design process, focusing on such things as budgets, costs, expenditures, schedules, and technical performance.



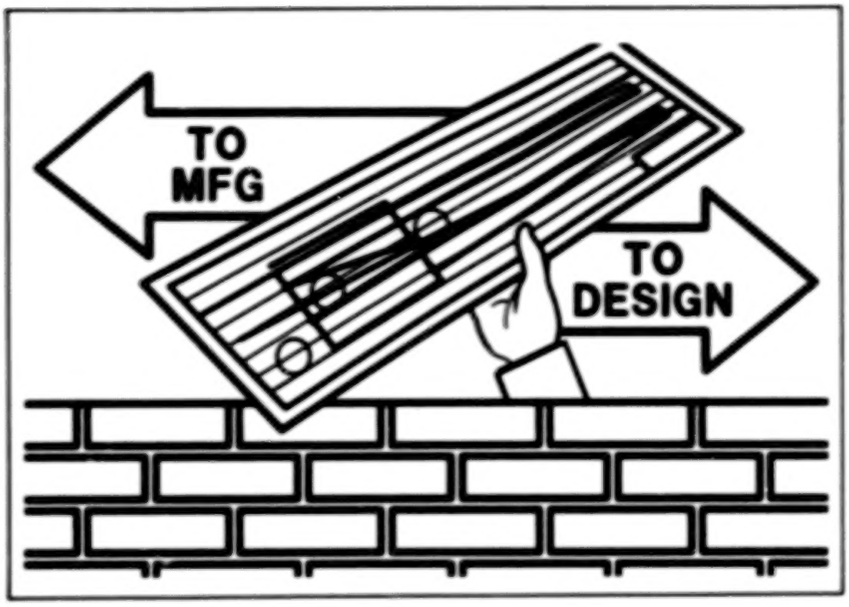
General design support is that set of data, precedent, and tools available to the design engineer. It has been estimated that two-thirds of the design engineer's time is used in gathering data needed to perform the design tasks—the technical data that establishes constraints and requirements of design and geometry describing product configuration. Design, an iterative process, involves the proposal, analysis, and review of a configured product. Technical analysis is the process of evaluating proposed designs, often resulting in revisions of the proposed design. Computer-aided design systems are computer systems supporting the analysis and configuration control aspects of design through "conversational" interaction with the designer and the creation of geometry and associated data. General design support also includes the knowledge, experience, techniques, and methods of the engineer augmented by design directives and standards.



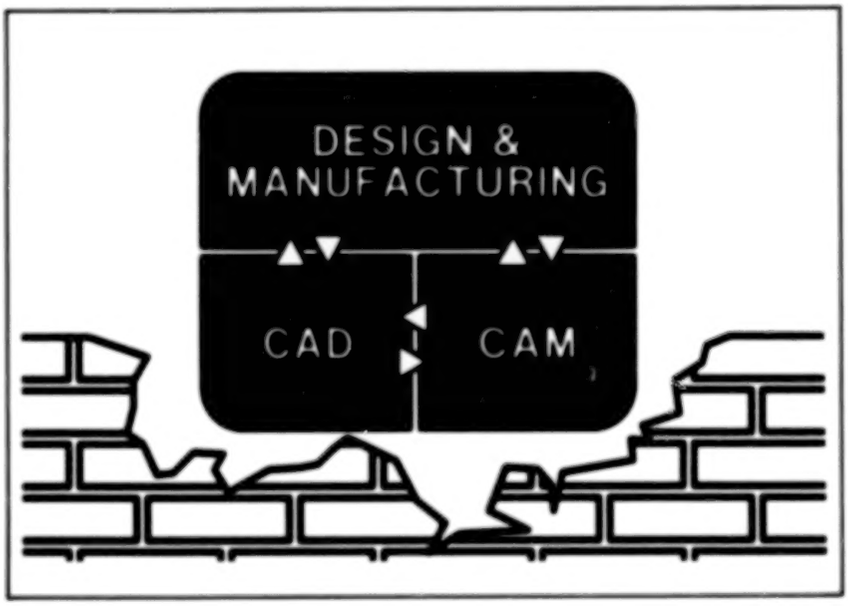
The third element of the design support foundation comprises the design network—the tapestry of diverse disciplines and activities woven together in a complex fabric of creation, analysis, and review. This network is a product of the evolving aerospace industry, based on historical precedent and reflecting a traditional design technology. The advent of computer-assisted design has dictated a re-evaluation of these traditional methods of conducting design and has resulted in a fundamentally more efficient and productive process.



Clearly, the design process as depicted here is extremely complex, requiring an enormous amount of managerial, organizational, and interdisciplinary communication. The design engineer sitting at his draftboard or CRT and working on his particular package is unaware of the overall environment he is working in, much the way an individual cell in the human body serves a specific function without being aware of the organism of which it is part.

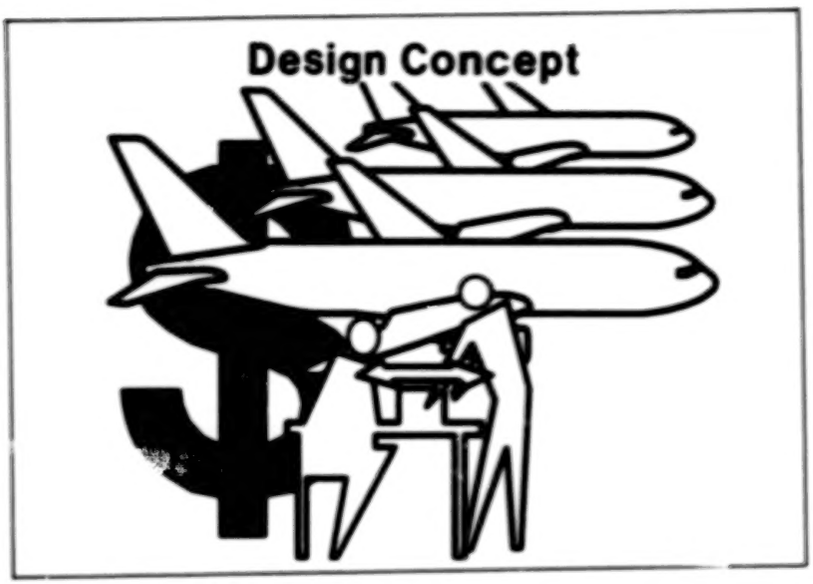


As previously noted, design and manufacturing used to be considered separate processes. After Design "threw the drawings over the wall," Manufacturing built the part with no other constraint than the drawing—the only contact with Design. As long as the form, fit, and function of the finished part satisfied the engineering drawing, all was well.

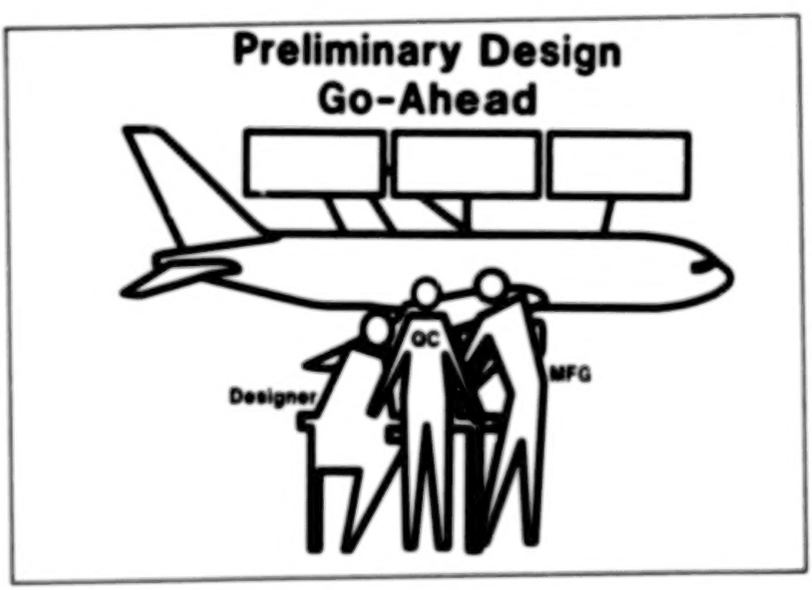


...Today, the wall between the design and manufacturing process has come tumbling down. More and more, the emphasis in aerospace is on the integration of design and manufacturing.

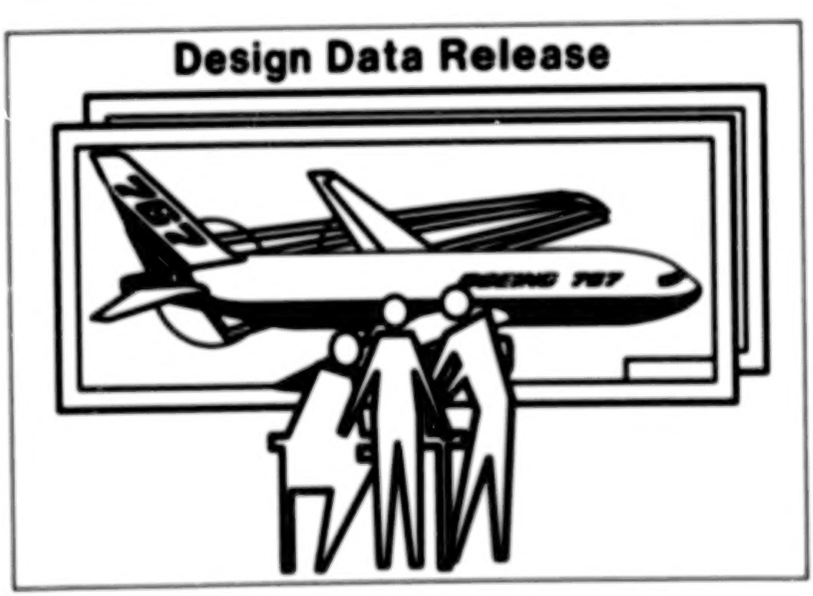
There are, of course, many powerful reasons for such integration; as usual, the computer is the logical device for achieving it. As aerospace vehicle designs become increasingly complex and sophisticated, designers have found that their tasks would be impossible without CAD systems. Similarly, computer-assisted manufacturing (CAM) has proliferated in the factory. It is a natural and logical development for CAD and CAM systems to work directly together. Thus, by default, design and manufacturing are linked through CAD/CAM integration. The most fundamental force driving the integration is cost. In the modern aerospace environment, manufacturing and design interact throughout the design process; decisions made early in the design process can therefore affect later manufacturing costs in profound ways. By adapting design decisions to manufacturing cost realities, it is possible to achieve enormous savings without diminishing the integrity of the design.



During the conceptual design phase, manufacturing interactions usually occur between staffs rather than projects. This is because, unless the new product is a derivative of an existing model, the manufacturing project is not even organized until preliminary design has been given a go-ahead; therefore, interactions during this phase primarily provide information to support high-level decision-making related to broad manufacturing capabilities and related costs.



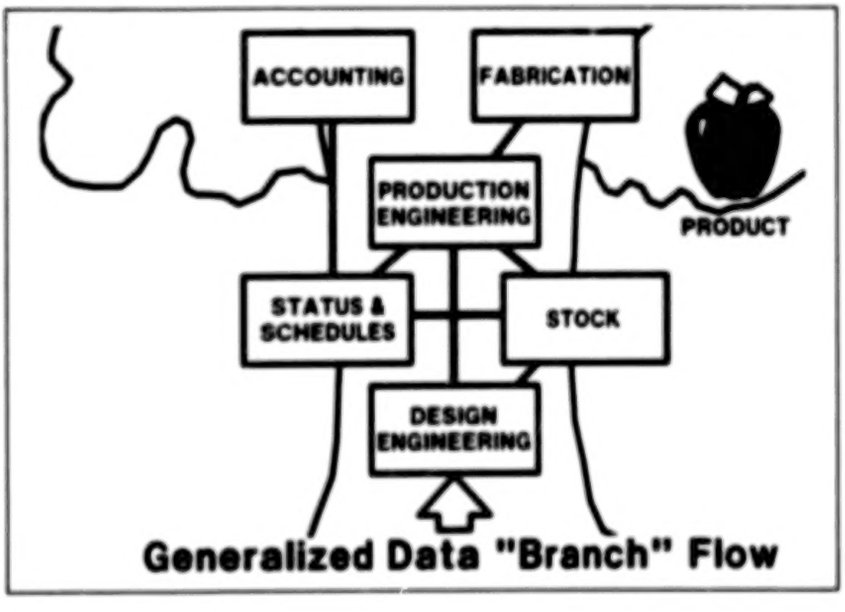
The preliminary design go-ahead triggers the Manufacturing and Quality Control organizations to establish projects to support product fabrication. These are initially skeletal organizations that will build up for the subsequent detail design and fabrication phases. Information developed for the preliminary design phase will be a refinement of the conceptual design phase based on more detailed and complete information.



After critical design reviews of the product have been passed by Design and Manufacturing, the go-ahead is given for the release of design data to Manufacturing. In the detail design phase, the design is refined and the details and components are designed in preparation for release. Manufacturing and Quality Control refine their plans based on more complete design data.

The fabrication phase begins as the first design releases become available and continues as long as the product is being built and delivered. Design changes due to engineering improvements, corrections, and manufacturing problems sustain the engineering design and manufacturing interface throughout this production phase. This interface reflects a design engineering bias that is quite common in aerospace.

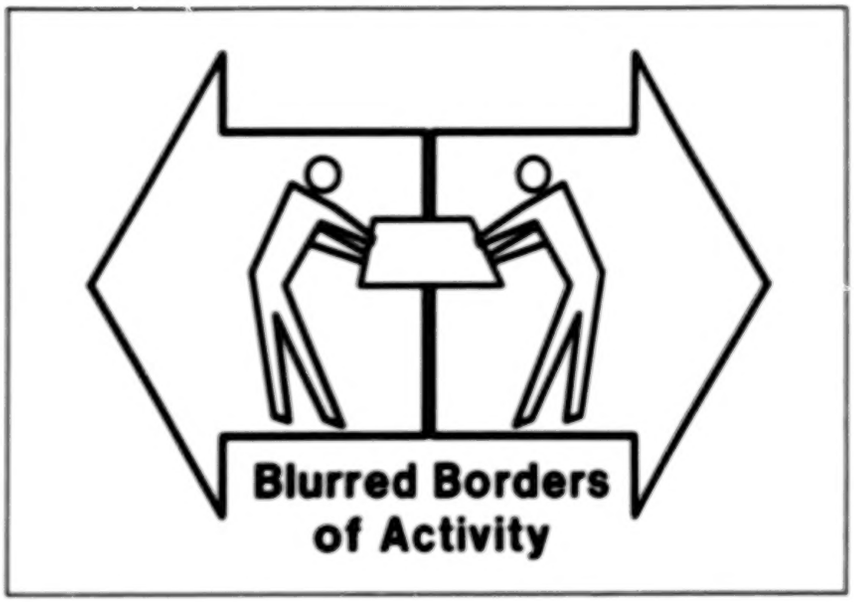
...From the manufacturing point of view, however, design is a manufacturing support function, and design is viewed as a source of data about the product. From this point of view, design is a subprocess of manufacturing!



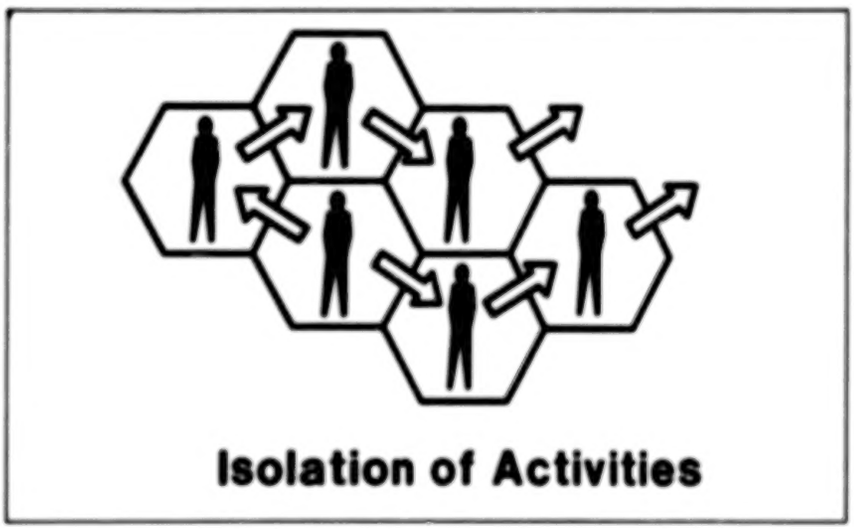
...Six activities supporting the manufacturing process are *design engineering, production engineering, factory operations, status and schedules monitoring, stock maintenance, and finance accounting*. These six categories illustrate how data is used in one particular aerospace company. It is relevant to discuss the nature of the problems encountered in the manufacturing process in terms of data flow because any manufacturer who attempts to integrate design and manufacturing must first define the data requiring processing and where it is to be used downstream.

It is critical that, while the manufacturing process is being automated, inherent inefficiencies, redundancies, and errors are discovered and eliminated, not simply hardcoded into computerized data management systems. Fortunately, the computer will not merely speed up current methods but will force a complete re-evaluation of the process.

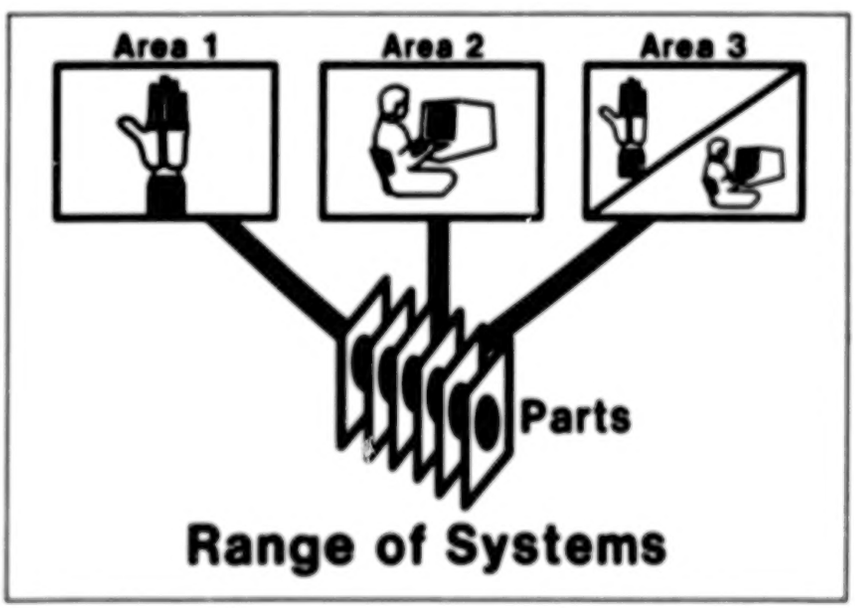
Typical of the problems faced in defining data and data flow are the following.



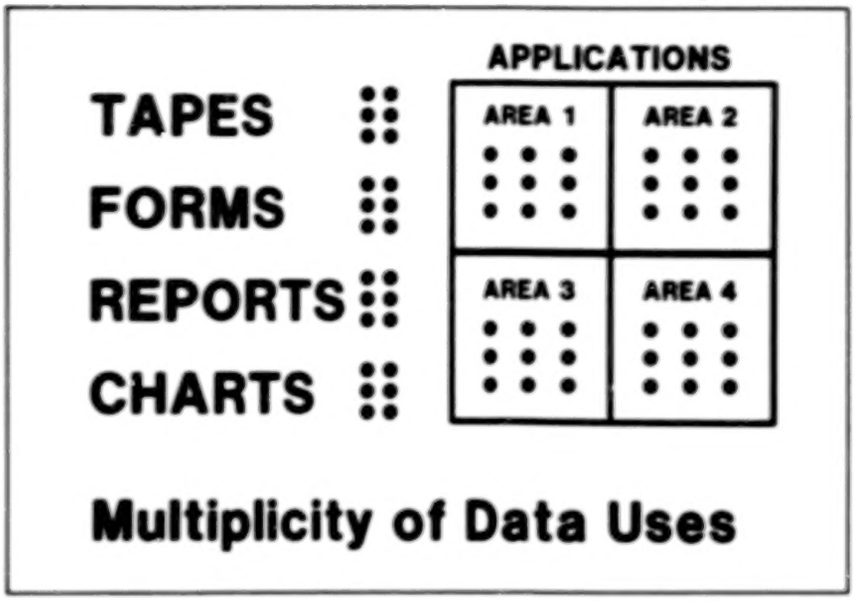
Blurred borders of activity. Many activities involved in the manufacturing process are not clear-cut but tend to overlap, especially those that are remote from the shop floor.



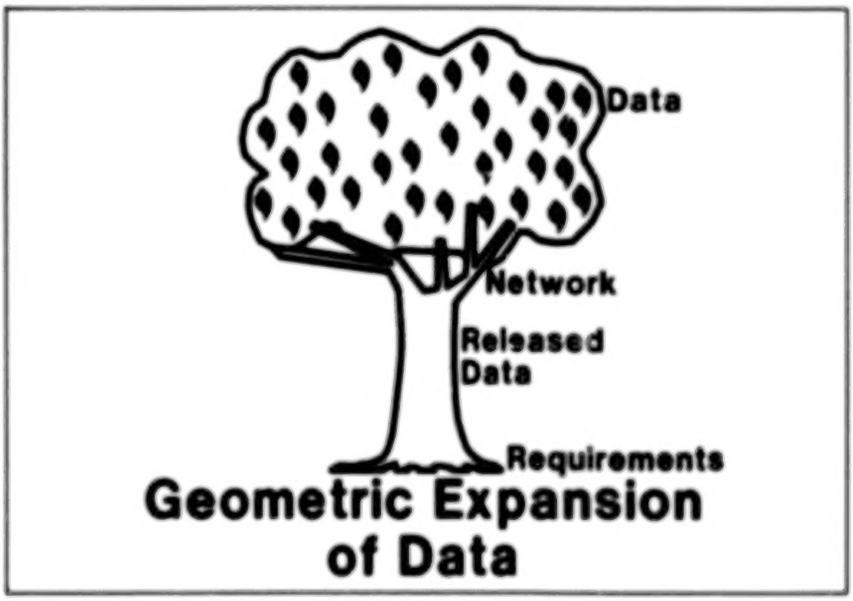
Isolation of activities. This compounds the overlapping problem because each activity has viewed the data in terms of input and output.



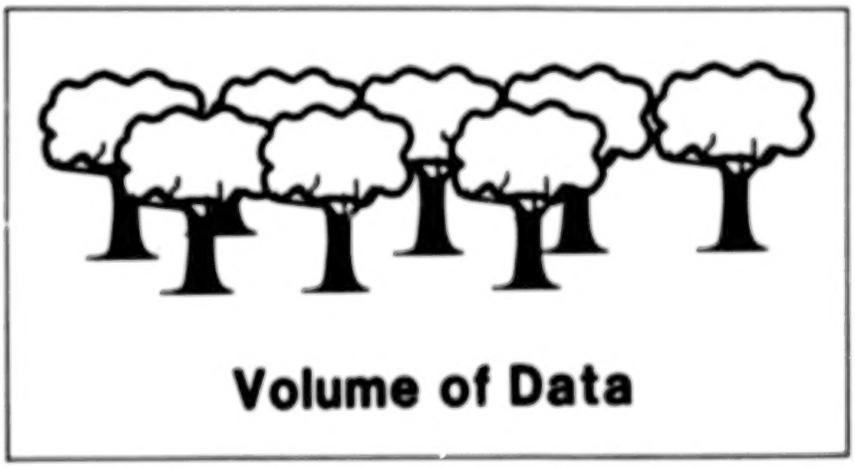
Range of systems within a company. The proliferation of computer systems throughout a company makes the overall task of data analysis difficult. Typically, the manufacturing data is a blend of manual, semi-automated and automated systems. The documentation associated with these systems suffers from a lack of standards and consistency.



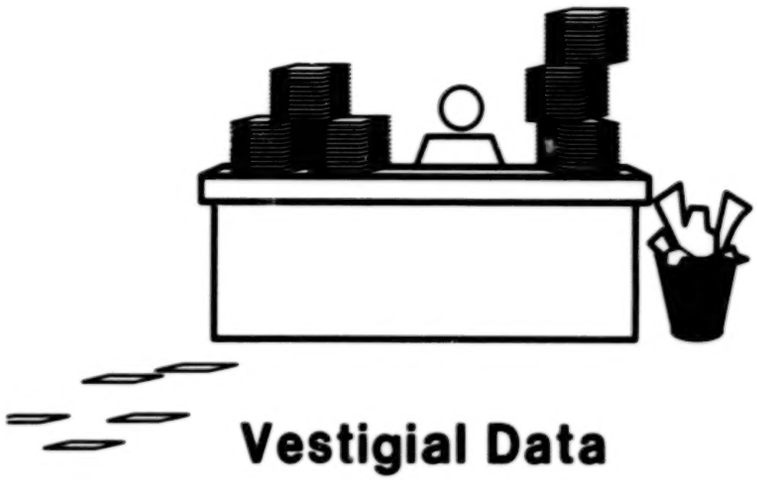
Multiplicity of data uses. A given data module (form, chart, report, computer tape, etc.) can be used by activities in different ways. For example, a form can serve as a means of data transmission with each activity inserting, extracting, or modifying data as required.



Geometric expansion of data. As data flows through the manufacturing system, it branches out. The paths of a really significant data item are indescribably complex.



Volume of data. The manufacturing process demands large volumes of continually updated data.



Vestigial Data

Redundant data. Data is used imperfectly. Typically, the data module repeats data available in other data modules or, in some cases, in itself.

Extraneous data. A data module can contain data that is not required for the activity it supports.

Vestigial data. Since the manufacturing process is continually evolving, the data required to support it is also evolving. As a result, a stored data item may be the remnant of prior methods.

It is apparent from these examples that the task of defining the data required in a company's manufacturing processes is a difficult and complex one but necessary if manufacturing data is to be managed and integrated.

The Manufacturing Process Foundation

- Management
- Support Groups
- Network

Data Base Management

The foundation supporting the manufacturing process is similar to that of the design process. It consists of manufacturing management, manufacturing support groups, and a manufacturing network. Since this concept has already been discussed, it need not be repeated. Rather, let us look at the data base management requirements necessary to support the manufacturing process. After defining the component activities of manufacturing, the data management requirements for those activities can be compared with those of design. Let me first review the major activities that comprise aerospace manufacturing and the tasks identified with each activity.

Production Engineering

- Manufacturing Plan
- Tool Design
- Quality Control
- Manufacturing Standards

The *production engineering activity* comprises four major tasks:

1. Plan manufacturing: prepare and maintain all tooling and production plans and test orders.
2. Tool design: design tools and fixtures required to produce the part.
3. Assure planning quality: review all tool and production plans, test orders, and revisions for adherence to quality standards.
4. Release manufacturing standards: prepare and release craftsmanship guidelines.

Fabrication

- Manufacture
- Quality Control
- Document Product
- Maintain Inventory

The *fabrication activity* consists of four major tasks:

1. Manufacture product: fabricate tools and products.
2. Assure product quality: assure that fabrication of tools and products meets guidelines for the craftsmanship of products.
3. Document product: gather all the documents that pertain to product geometry and fabrication.
4. Maintain inventory: maintain all inventory in fabrication stores.

Status and Schedule

- Maintain Schedules
- Forecast Requirements
- Control Inventory
- Product Visibility

The *status and schedule activity* involves the following tasks:

1. Maintain schedules: prepare and maintain master schedules, part schedules, and shop schedules.
2. Forecast requirements: project future work orders by analyzing schedule data, product history, seasonal requirements, and sales forecasts.
3. Control inventory: maintain and control the location of all stored inventory.
4. Report status and requirements: maintain and locate all inventory including stores and in-work inventory.

Stock

- Select Vendors
- Purchase Materials
- Receive & Inspect
- Maintain Inventory

The *stock activity* comprises the following tasks:

1. Select vendors: determine vendors having materials and/or parts needed to support scheduled fabrication.
2. Purchase material: prepare a purchase order from data on a purchase request form.
3. Receive and inspect: accept and test all incoming vendor parts.
4. Maintain inventory: determine status, location, and quality of all the stored material not located in the fabrication area.

Accounting

- Accumulate Costs
- Generate Reports
- Maintain Records

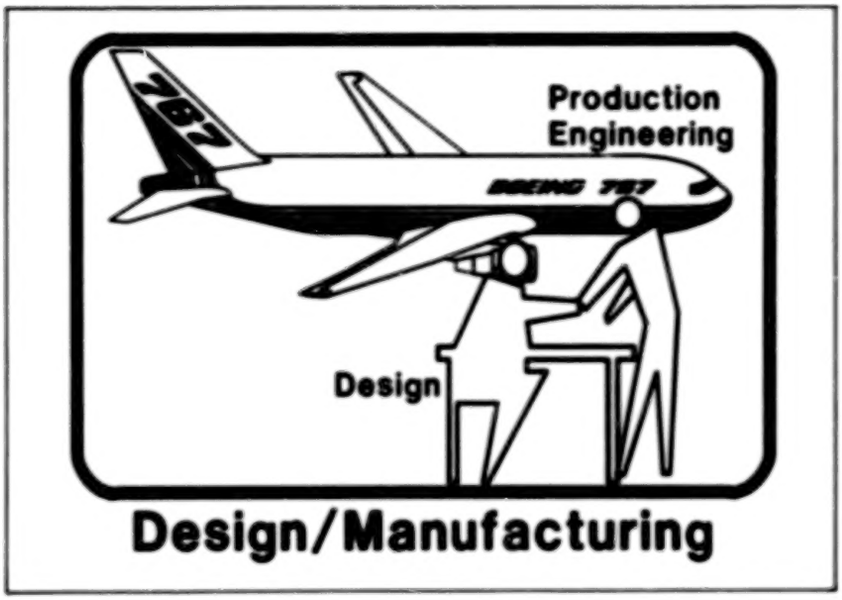
The *accounting activity* consists of the following tasks:

1. Accumulate costs: gather all the manufacturing costs including material, labor, facilities, and depreciation for use in analyzing the product budget.
2. Generate account reports: prepare reports on the product budget and other financial activities.
3. Maintain product records: determine location and status of the completed product's documentation.

- Design Engineering
- Production Engineering
- Fabrication
- Status & Scheduling
- Inventory
- Accounting

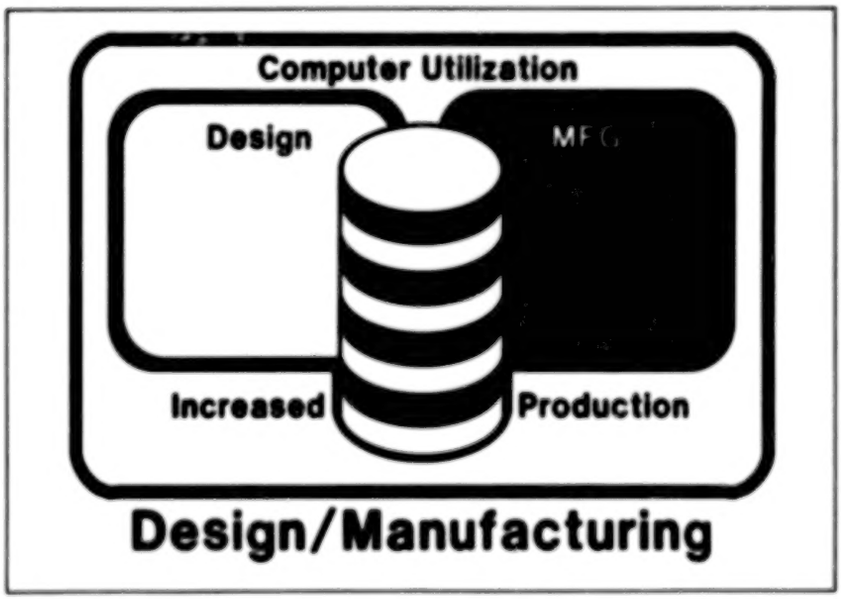
Data Base Management

These activities and tasks, though organized differently from company to company, are all required no matter how the department is arranged and are linked together by the flow of the data network. To integrate them, the data base manager must respond to the particular needs of each activity. The requirements of a data base manager to support the entire design and manufacturing environment will be enumerated later in this presentation.

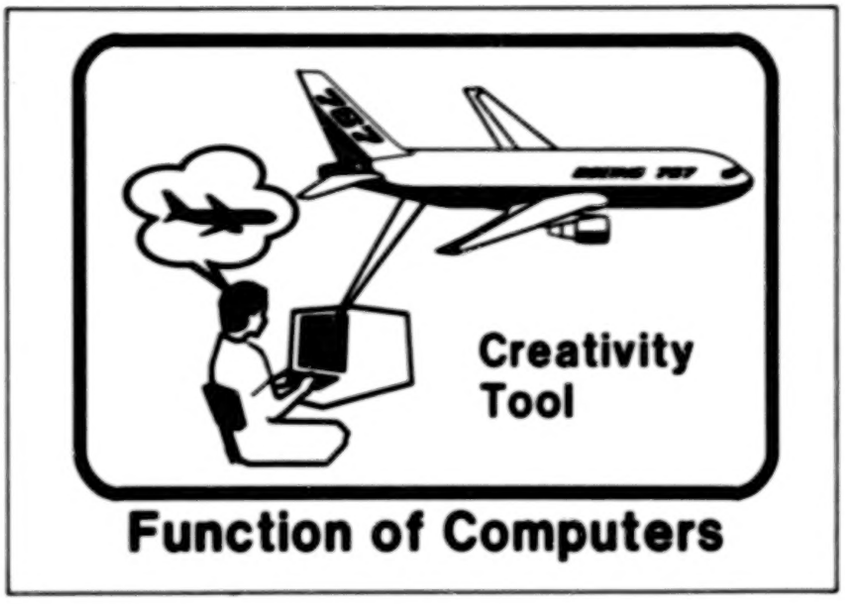


DESIGN VERSUS MANUFACTURING CHARACTERISTICS

The design process must incorporate aspects of design that directly impact the cost and producibility of a product. Historically, production engineering design of tools and numerical control programs, even though functionally related to engineering design, has been treated as manufacturing because those functions determine how a part is to be built. When viewed from the perspective of the data function, production engineering and design engineering are even more closely aligned because the data items shared are the same: geometry and related information.

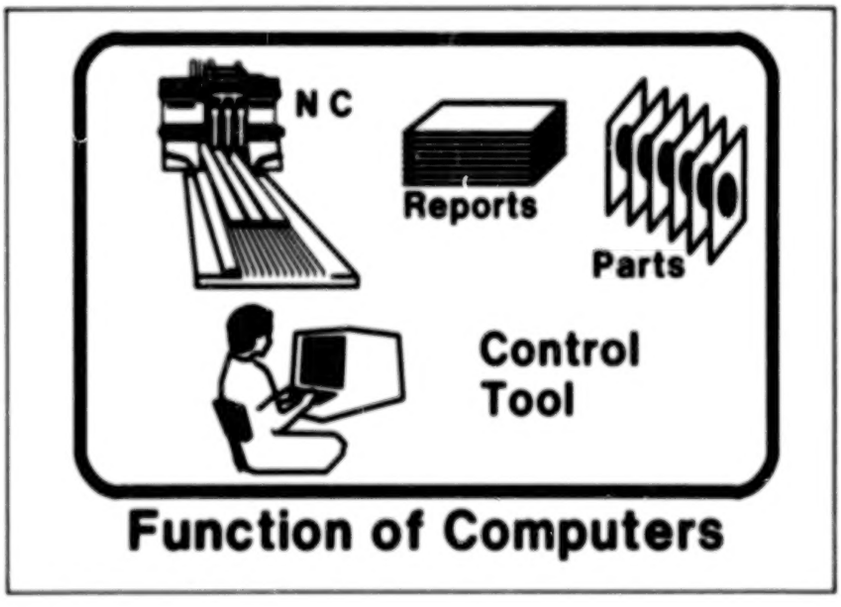


Another perspective bearing on the relationship of the manufacturing and design disciplines has to do with how computers are used. If quasi-design functions currently existing in production engineering are removed from manufacturing, a natural delineation between manufacturing and design becomes apparent. On one side, we find functions that can be characterized as part design/analysis (which are referred to hereafter as design); on the other side, we find part accounting/control, which is usually associated with factory-floor level and supporting organizations and hereafter will be called manufacturing. Of course, these distinctions do not hold true in a pure sense in all aerospace corporations; in an environment as dynamic as design and manufacturing, exceptions to these generalizations must be expected. This does not mean that the characterizations are invalid, only that they must be qualified. Also, it should be kept in mind that computers serve both manufacturing and design to achieve a common goal—increased productivity. Thus, in the myriad ways in which computers are used to increase user efficiency, design and manufacturing have more in common than the following comparison implies.

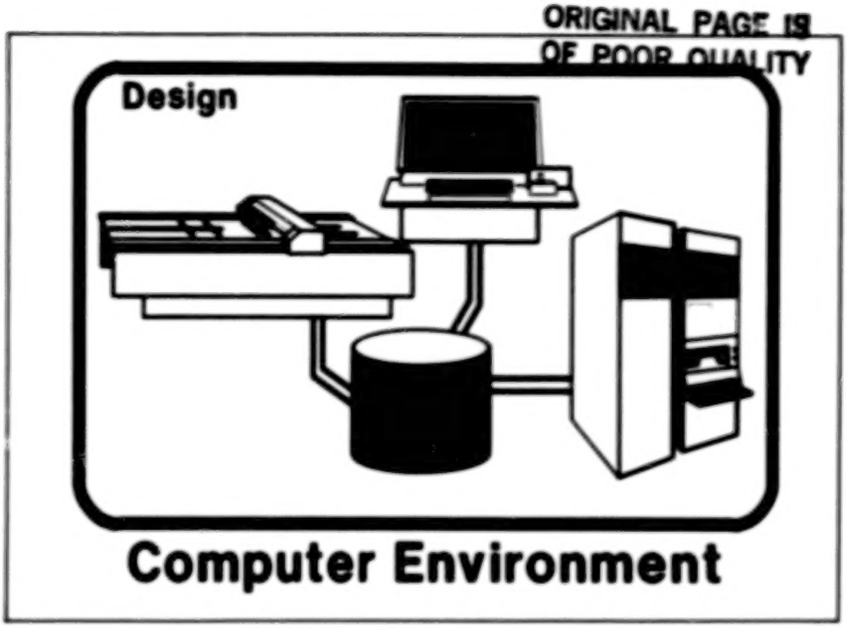


FUNCTION OF COMPUTERS

Design. The design process begins with the conception of an idea for an object and continues until it has completely defined that object. It is estimated that more than two-thirds of a design engineer's time is used in gathering data to perform the design task, which consists largely of creating geometry, performing analysis, and documenting configuration. The computer as a design tool assists the process as an information resource, analysis mechanism, and device for manipulating geometric entities—functions that once were supported by a library, a slide rule, a drafting board, and a pencil. These tools aid the design process in similar ways: they help the user not only to be more productive but more creative as well, enabling designers and analysts to explore new avenues of thought. Although this concept is somewhat idealized (designers may be assigned a heavier workload in lieu of exploring new concepts), the computer nevertheless becomes an aid in the process.

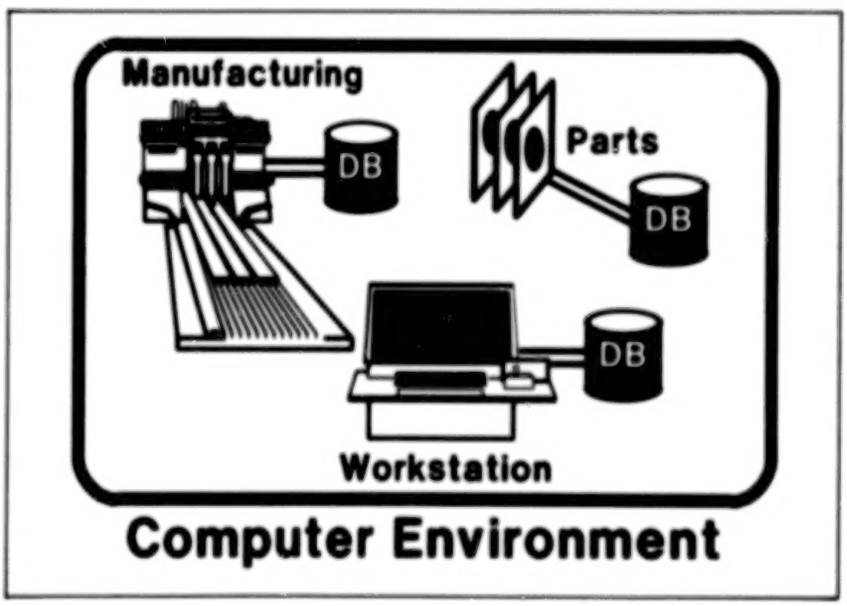


Manufacturing. Over the last few years computers have invaded the manufacturing process at an accelerated rate. Industrial robots, controlled by computers, have joined numerically controlled machines and material management systems in ever-increasing numbers on the shop floor. Factory resources, too, are coming under the control of computers that manage labor, accounting, shop loads, and many factory support services. Here, the computers serve mainly as controls. Direct control of the factory through a closed-loop feedback network of computers has been achieved in small firms and is nearly within reach of larger, more diverse operations. This would be impossible without the multiplicity of systems whose common denominator is computer control.

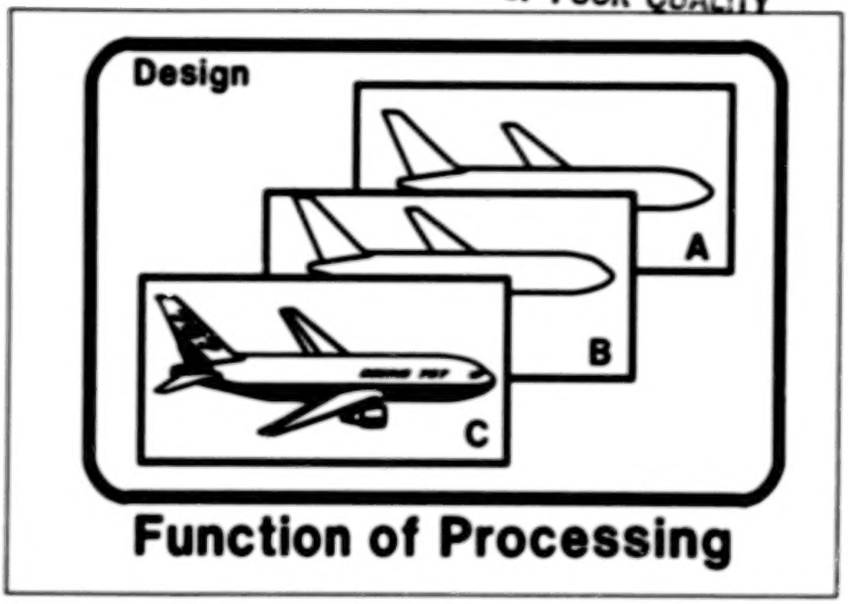


COMPUTER HARDWARE ENVIRONMENT

Design. The first application of computers in design in the late 1950's was in support of "number crunching" by large, general-purpose, mainframe computers located in dedicated facilities isolated from users (ref. 5). Later, true computer-aided design (CAD) systems were developed and used as simple, automated plotting devices. As CAD technologies developed, other systems were added, bringing computing resources physically closer to users. CAD minicomputers and desk-top microcomputers serving as independent work stations added new dimensions to the design process. Computer hardware used in design is now a heterogeneous blend of single-user, standalone, turnkey, CAD systems that can be integrated to share and perform operations on the same data. More typically, however, these systems are interfaced, making it possible to transfer data from one system to another electronically. Often these interfaced systems must contain intermediate programs that format the data in order to be compatible with each other.



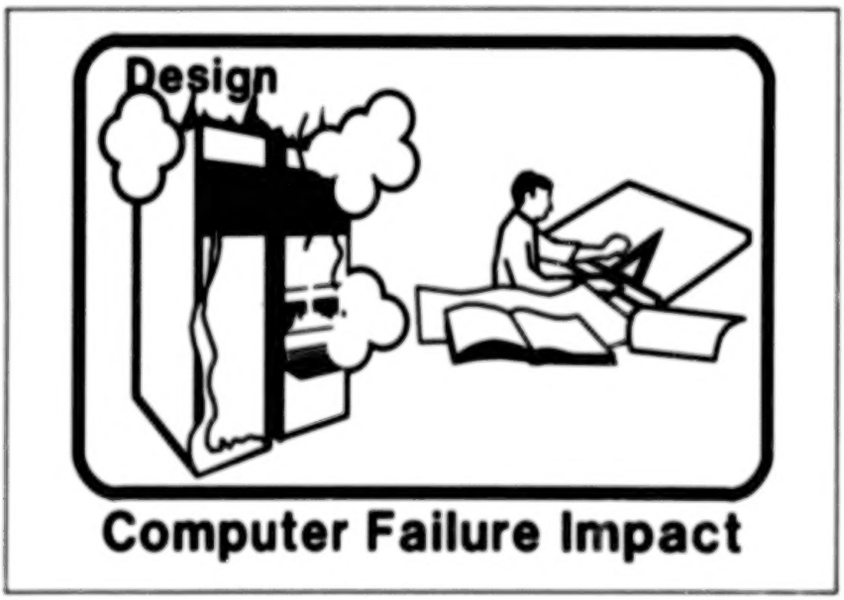
Manufacturing. The application of computers to manufacturing has been a fairly recent phenomenon. In its initial phase, the computer-aided manufacturing (CAM) system used automated machinery to transport material and perform simple mechanical operations. Today, these systems have achieved greater levels of sophistication in complex processes. The computer, although the controlling mechanism in NC machines and robots, is not routinely accessed by shop personnel, but its impact is felt by all in material and resource accountability, tracking, and status reporting. The CRT terminal and other computer input devices have become standard shop hardware.



FUNCTION OF PROCESSING

Design. The design process is highly iterative; indeed, a design is never complete in that another iteration is always possible, with some benefit invariably derived. Since each iteration can create a new data version, the data can be organized into logical sets, each of which is treated as a functional unit by the designer. A data set, for example, can define a part geometry or contain finite-element analysis matrices. Data set versions allow for the update of complex information in a logical, controlled manner. From the computer processor's view, design uses computers to version data sets. Often a history of data set versions is maintained to back up design decisions and permit recovery of previous work.

Manufacturing. Unlike design, the production process, at the shop level, is interested in whether an event has occurred or not. Updating of information concerns status of material, factory loads, personnel, shop operations, etc. Typically, the updated item is a single data element or field that requires update and is not CPU-intensive.



TYPE OF PROCESSING - UPDATE CYCLE - BACKUP AND RECOVERY

Design. The design process uses computers to process data in a variety of ways. Large analytic programs are run in the batch mode; CAD systems run interactively. Although driven by milestones, design is not fundamentally real-time dependent. Because of the volume of data associated with data set versions that require interpretation and analysis, daily iteration is usually acceptable. Response cycles can range from a few seconds to a few days. Since design uses data set versions, backup and recovery can be accomplished smoothly following system crashes. As long as previous versions are archived or offloaded, no more than a few hours or a day's work need be lost; at worst, design can continue by means of previous manual methods. Thus, the CAD process is not unduly sensitive to time constraints and system crashes.

Computer Failure Impact

Manufacturing. In order to control shop activities, which are inherently dynamic, the status of all shop resources must be continuously updated and each change recorded. The basic unit of interaction with a computer is the transaction, for example, "item 42, arrived station L11, 11:03 a.m.," which would be one transactional update on the record of item 42 or of station L11. Transaction processing is used by manufacturing under computer control. Correspondingly, the cycle time for updates is real time, so production delays and bottlenecks must be resolved quickly. Only real-time support systems can satisfactorily maintain the production flow in complex, interdependent manufacturing environments, where operating systems must be extremely sensitive to feedback from the shop floor to take advantage of this updating capability. Clearly, in the production environment, system crashes are a critical problem that must be effectively isolated.

Manufacturing Information Manager

MANUFACTURING INFORMATION MANAGER REQUIREMENTS

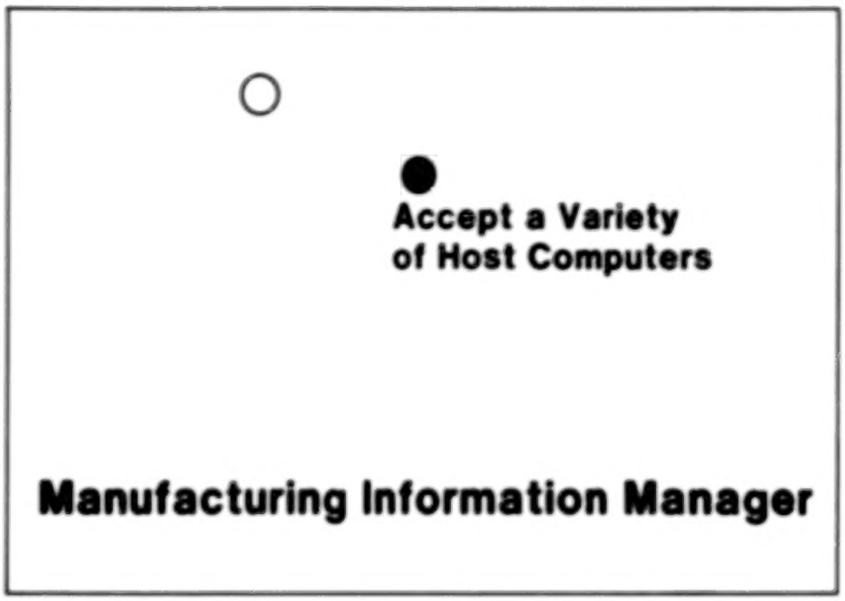
The IPAD requirements define the capabilities needed to support the design process. A manufacturing information manager (MIM) shares these and other requirements that are critical to the manufacturing environment. This section of the paper addresses requirements concerned with the integration and management of data that controls a typical manufacturing process.

MIM requirements do not dictate specific design criteria, nor do they estimate performance or acceptance criteria, which often lack meaning and depend on changing hardware technologies. The following are MIM requirements that address current and prospective manufacturing information management needs.

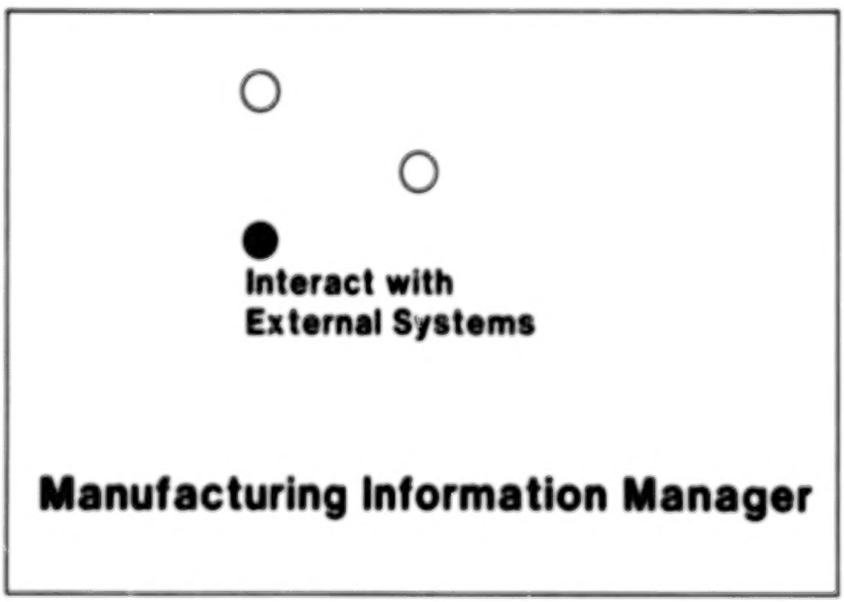
- Support Existing Computing Environments

Manufacturing Information Manager

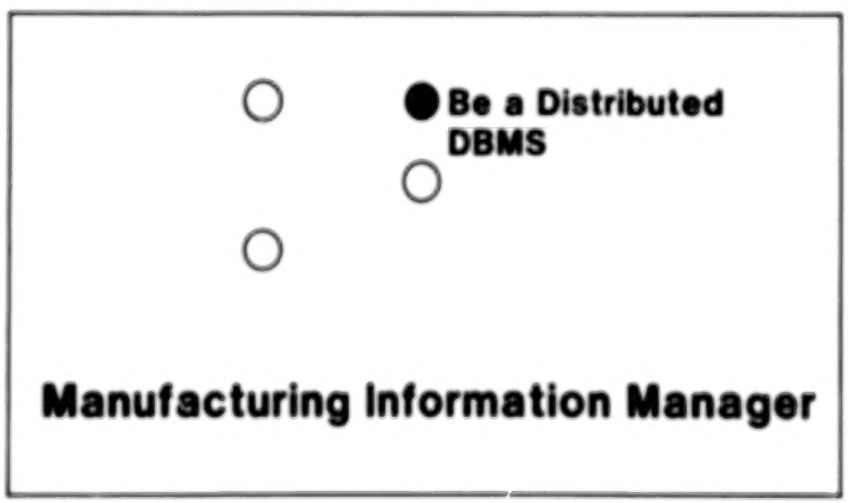
1. First, the MIM must support existing heterogeneous computing environments. This requirement is the foundation of a MIM system. A heterogeneous computing environment is one in which there are many kinds of computers from different vendors. These computers, in general, are independent of each other. The MIM is the means by which these diverse machines will be integrated, that is, function like an organic whole.



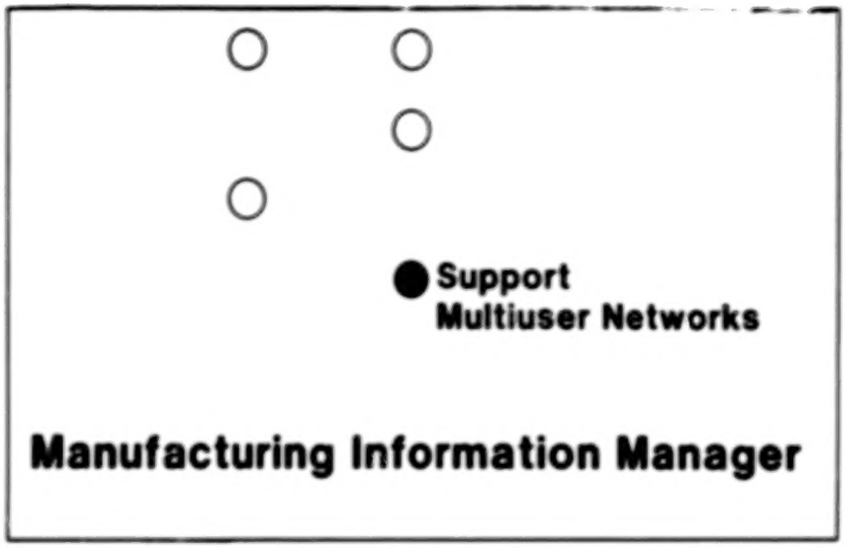
2. The MIM must accept a variety of host computers, not require a specific host machine.



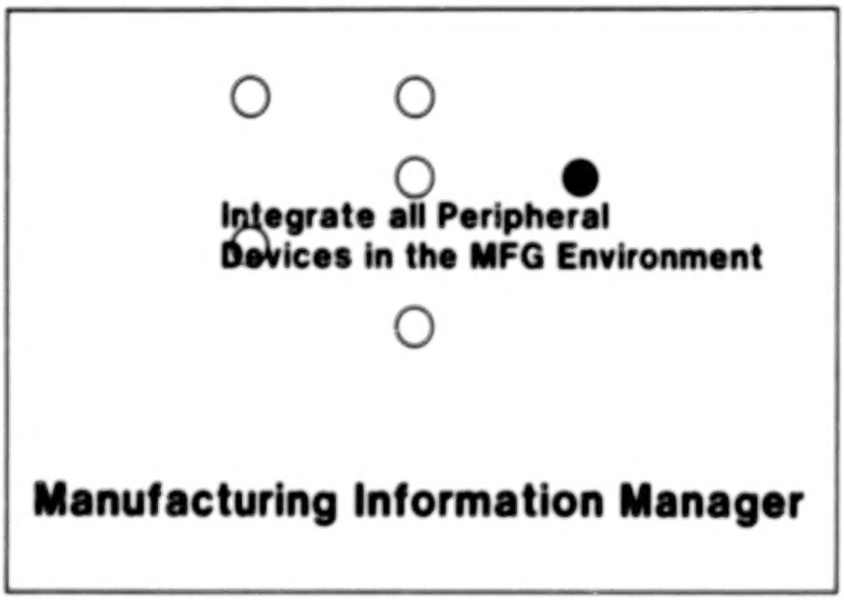
3. The MIM must interact with external systems. The integration function of the MIM requires that it be able to accept, process, and disperse data to external systems. It must have the capacity to act as a switchboard between computers using common interface standards.



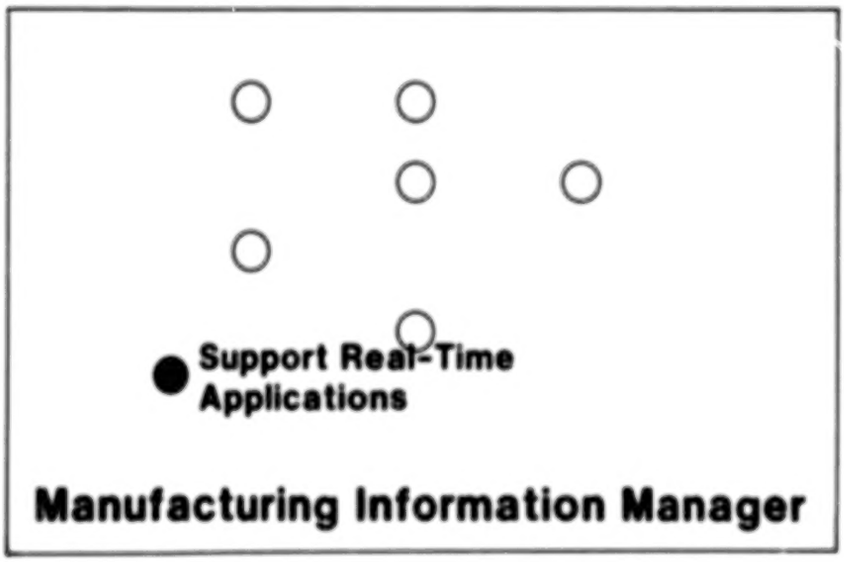
4. The MIM must be a distributed DBMS. To integrate information and functions in manufacturing, geographically dispersed machines must be able to share data and functions. The MIM must be able to utilize the data processing functions of existing machines as if they were MIM functions, and be able to use the data in those machines as if they resided on the host MIM machine.



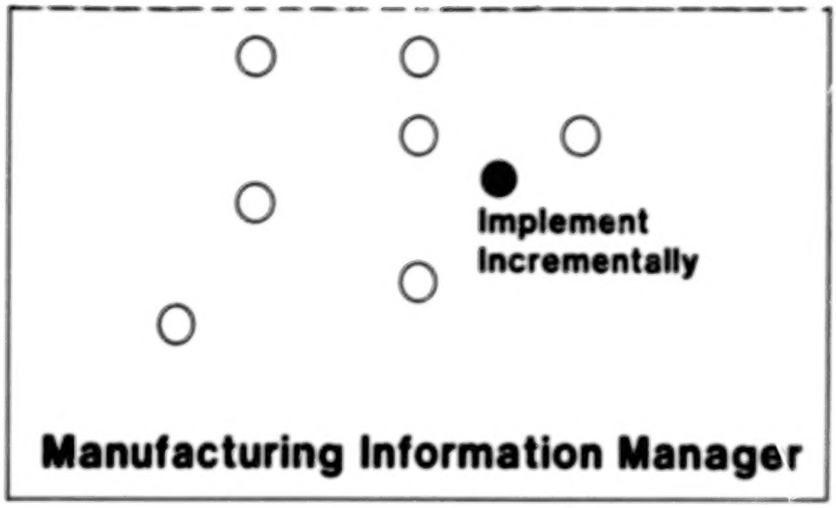
5. The MIM must support multi-user networks because manufacturing is a multi-user process. The integration of distributed machines requires a mechanism to interface these machines. The MIM must possess a facility to control the movement of data between machines in a multi-user network.



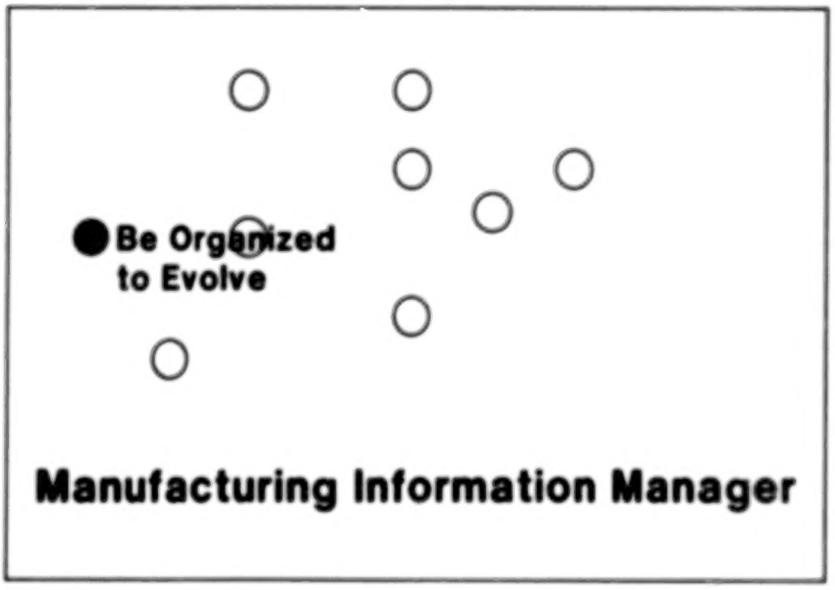
6. The MIM must integrate all peripheral devices in the manufacturing environment: terminals (smart and dumb), printers, plotters, light pens, bar code readers, portable microcomputers, and word processors to name only a few of the means by which data is created and displayed. The MIM must be able to utilize, directly and indirectly, all of the means available for input and output.



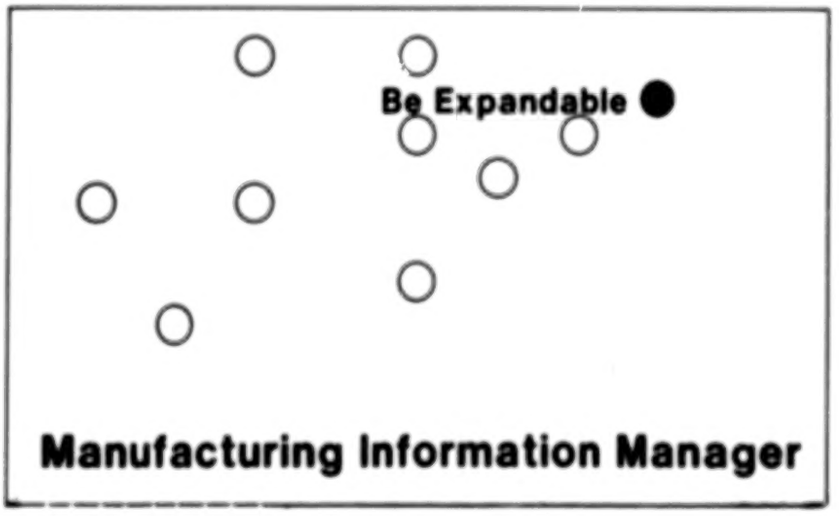
7. The MIM must support real-time applications. The direct control of machines in real-time is not the domain of the MIM. The control of real-time numerical control machines is the domain of dedicated local computers. The MIM must be capable of integrating local real-time systems into the manufacturing and design data base. In this way, geometry versions and associated data stored in the global MIM can directly initialize local computer-controlled manufacturing processes.



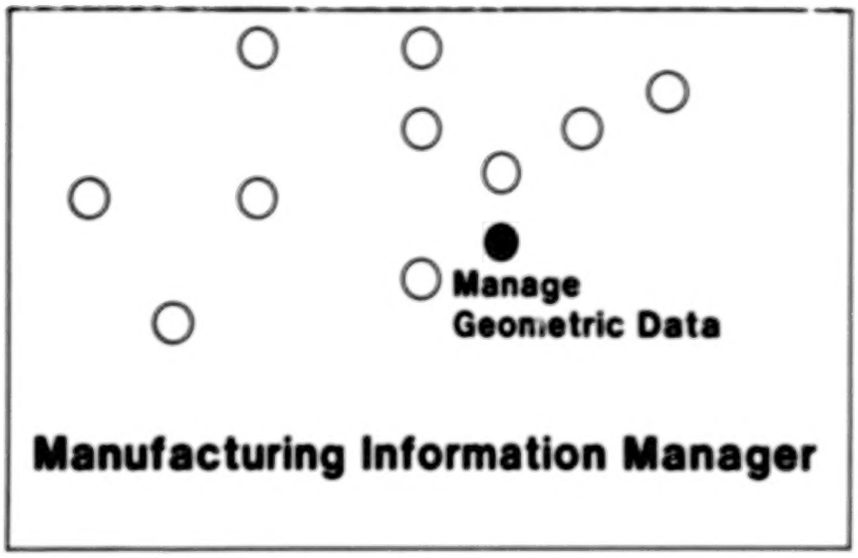
8. The MIM must implement incrementally, as manufacturers need to integrate existing islands of manufacturing technology. A MIM implementation must tread softly on ongoing manufacturing activities. Ideally, the transition to a MIM system is imperceptible in terms of system downtime and production disruptions.



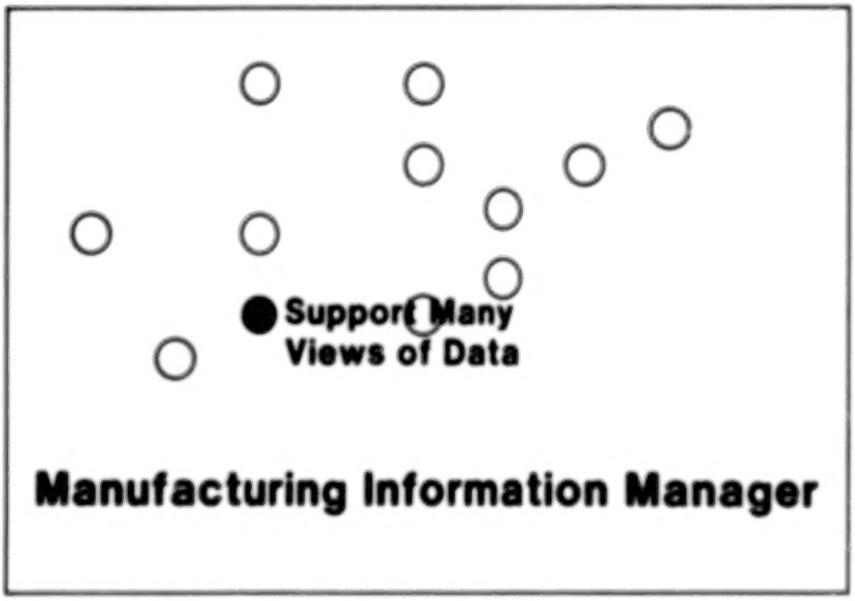
9. The MIM must be organized to evolve, that is, reflect in its own organization the dynamic nature of the processes it supports. The needs of manufacturing technology change; and MIM, though unable to anticipate the details of the change, must be flexible and easily modified.



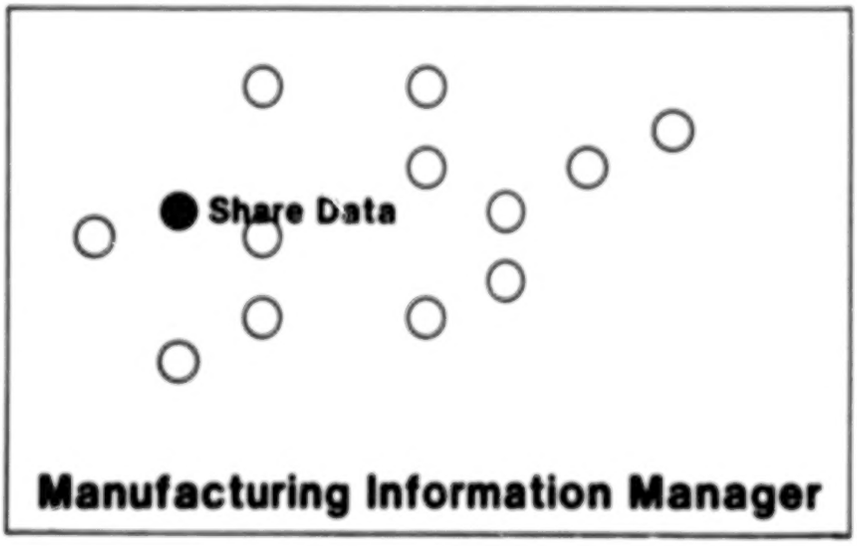
10. The MIM must be expandable, with no constraint on the kinds or quantities of hardware and software integrated into it.



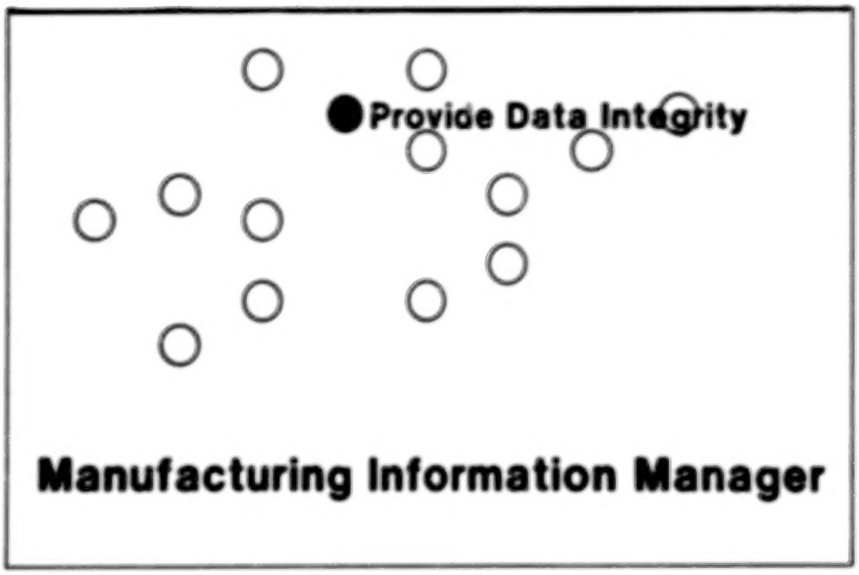
11. The MIM must manage geometric data, which permeates the manufacturing process. The configuration management of geometry is a major task confronting both design and manufacturing. There is need for a MIM that will automatically manage geometric and associated data.



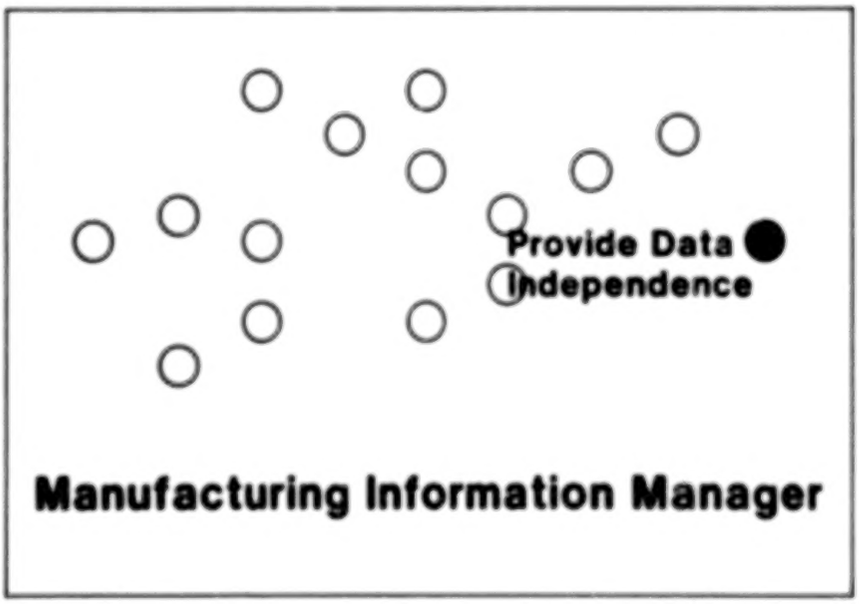
12. The MIM must support many views of data. Throughout the design and manufacturing process, data including geometry circulates through the system. Often information must be reformatted to satisfy the needs of a particular organization. A MIM must allow the user to access and modify data in a format that reflects the need of the activity, yet is consistent with the data source.



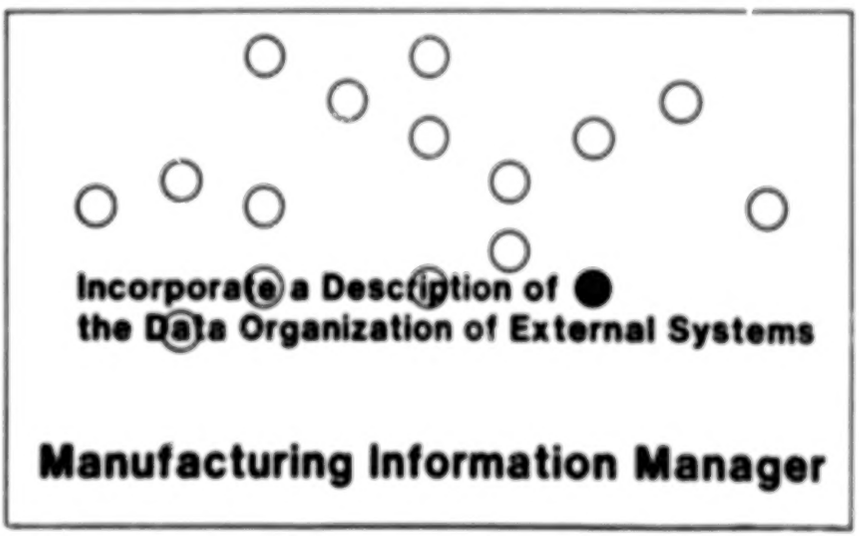
13. The MIM must share data, since manufacturing applications such as planning and material management are mutually dependent on common data. Often multiple copies of the same data physically reside on different machines or programs. This approach is inefficient and complicates the problem of data integrity: how to guarantee that the changes to these data have propagated to all copies. When data is shared, duplication is reduced and updates propagate uniformly.



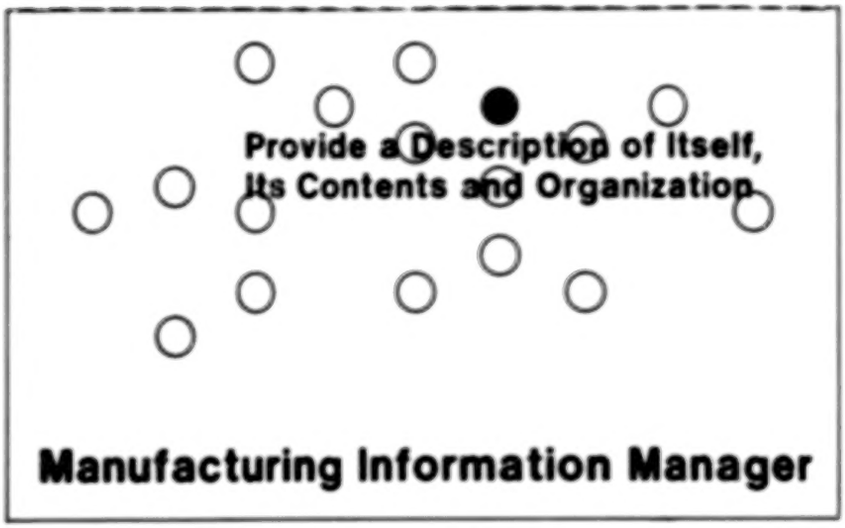
14. The MIM must provide data integrity, as external systems using a MIM will rely on it to access, store, and update data. It is the responsibility of the MIM to control the quality of these data, that is, its correctness based on constraint checking. Integrity is ensured by requiring data to qualify according to the enforcement of rules.



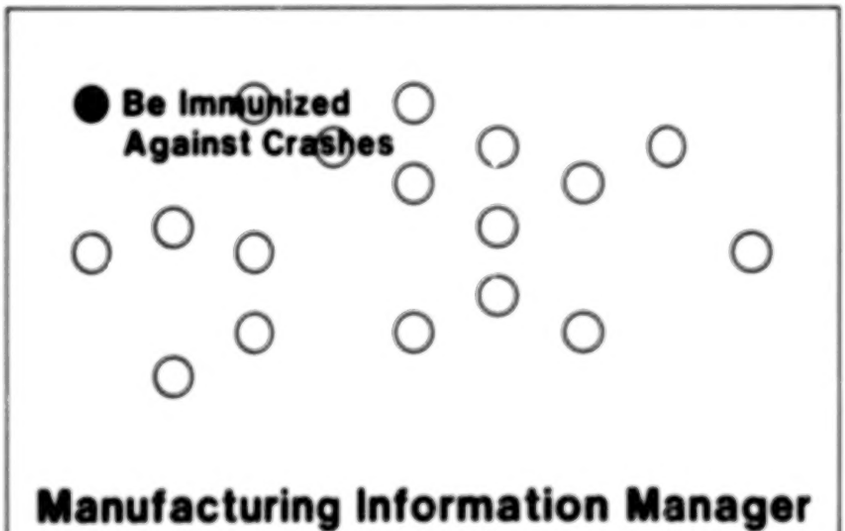
15. The MIM must provide data independence so that the constantly required changes to the data base that are inherent in the manufacturing process remain isolated from existing application programs to preclude the need to update the application each time the data base is changed.



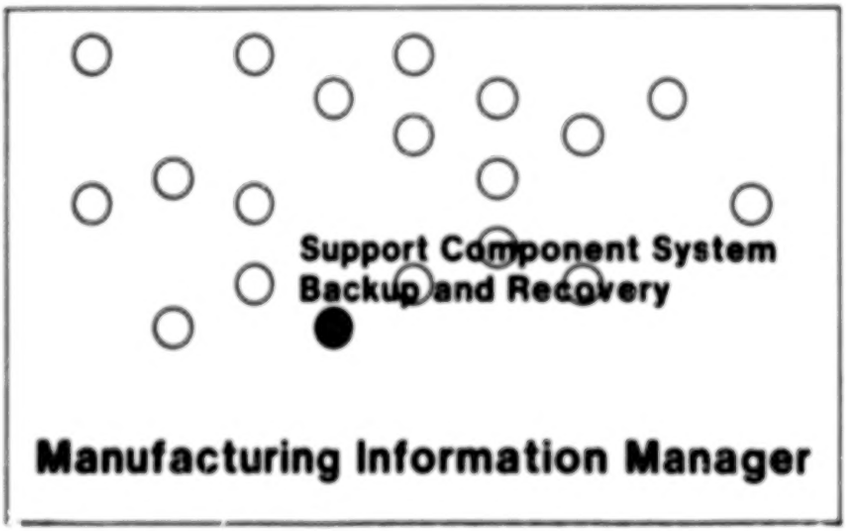
18. The MIM must incorporate a description of the data organization of external systems, that is, it must recognize how an external system organizes its data and must support the integration of that data model into its own.



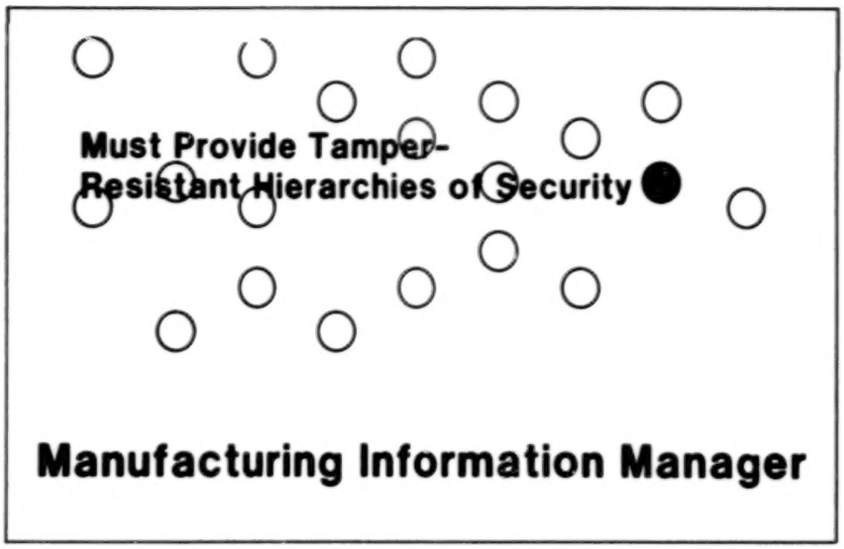
17. The MIM must provide a description of itself, its contents and organization to control its own evolution. (This information is commonly called meta-data and resides in a data dictionary.)



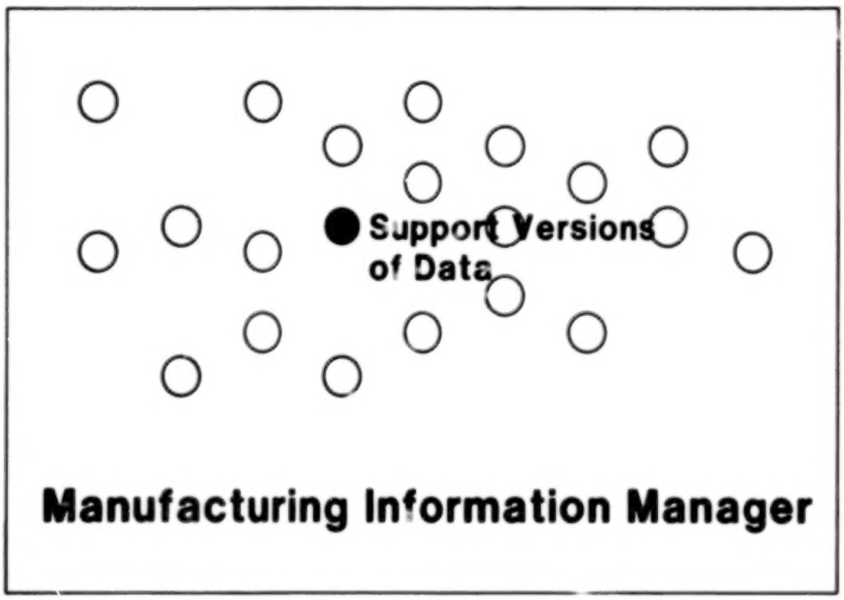
18. The MIM must be immunized against crashes, which are far more catastrophic in an integrated environment than in a local system, since a disfunction of the MIM can propagate throughout an entire integrated system.



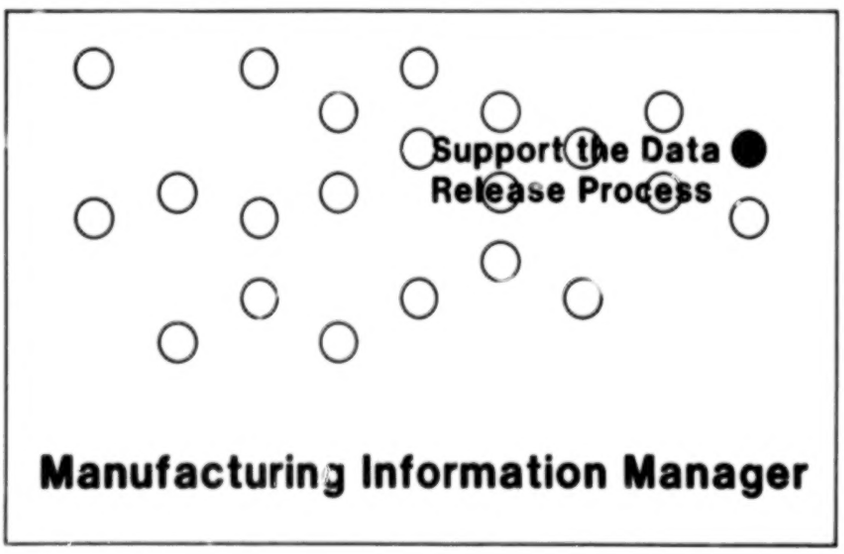
19. The MIM must support component system backup and recovery, isolating any failure of its external systems in order to limit the damage and coordinating the recovery of the failed system to ensure that it is synchronized with the integrated system.



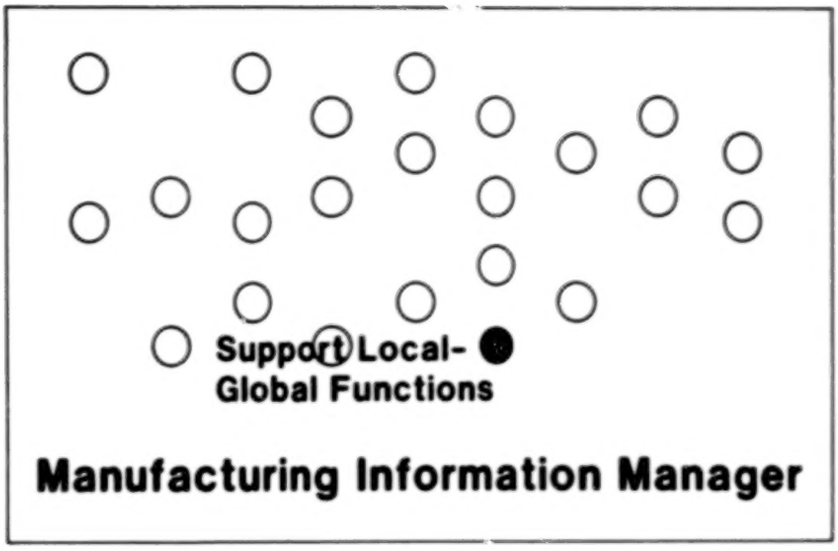
20. The MIM must provide tamper-resistant hierarchies of security, ensuring that security locks on external systems cannot be defeated. The host MIM must itself be secured from unauthorized use at all functional levels, and MIM must provide for the security of all the systems for which it is responsible.



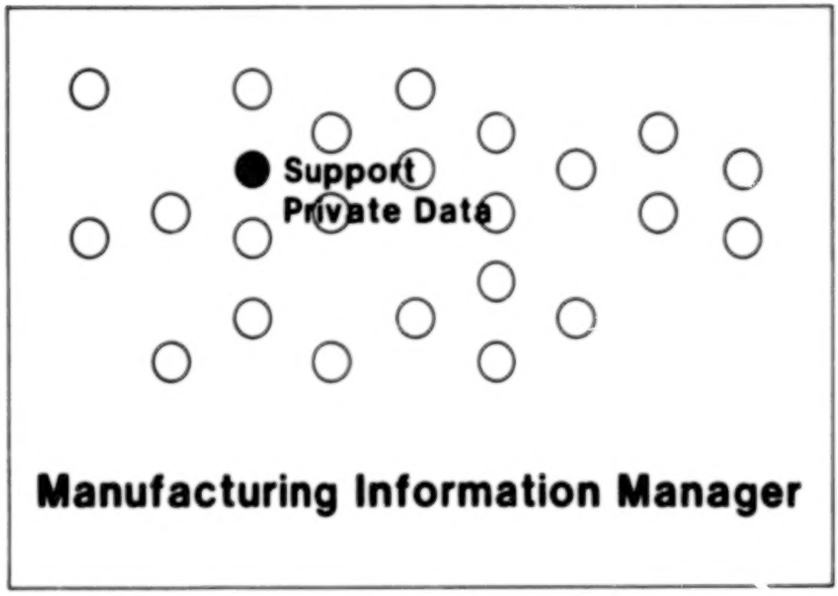
21. The MIM must support versions of data, specifically geometry, which is metamorphic: as it changes it becomes a new version.



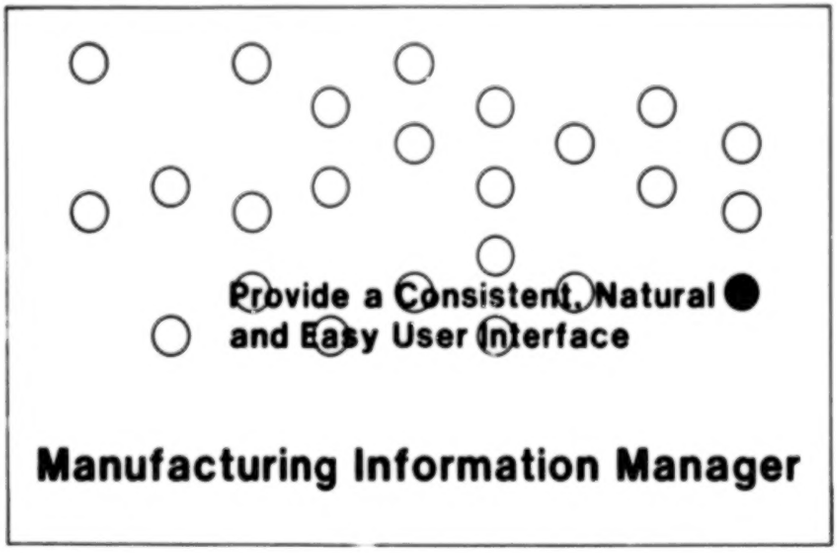
22. The MIM must support the data release process by providing a method of duplicating the signoff process used to guarantee the quality of data.



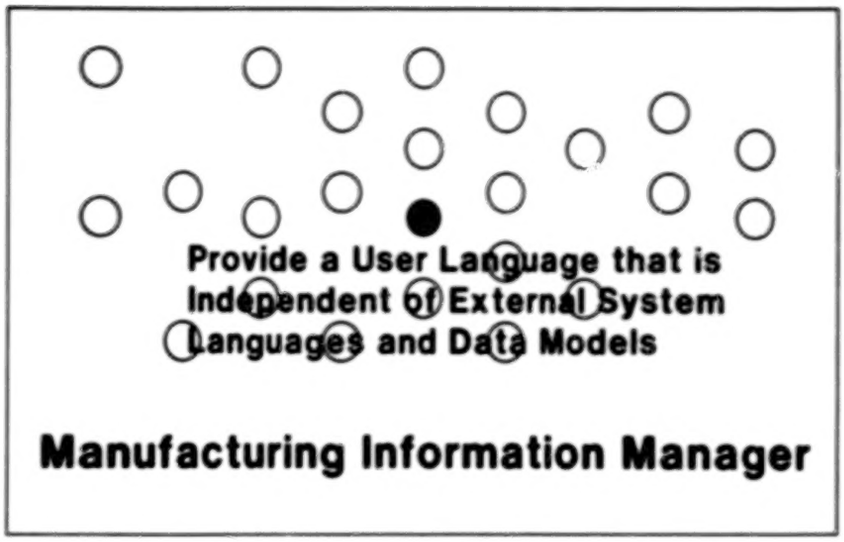
23. The MIM must support local-global functions, which can be interpreted as a hierarchy of control. Each controller takes commands from only one higher level while controlling several others at the next lower level. Data is filtered through these controls to a logical terminus, which may then execute the functions dependent upon the local data.



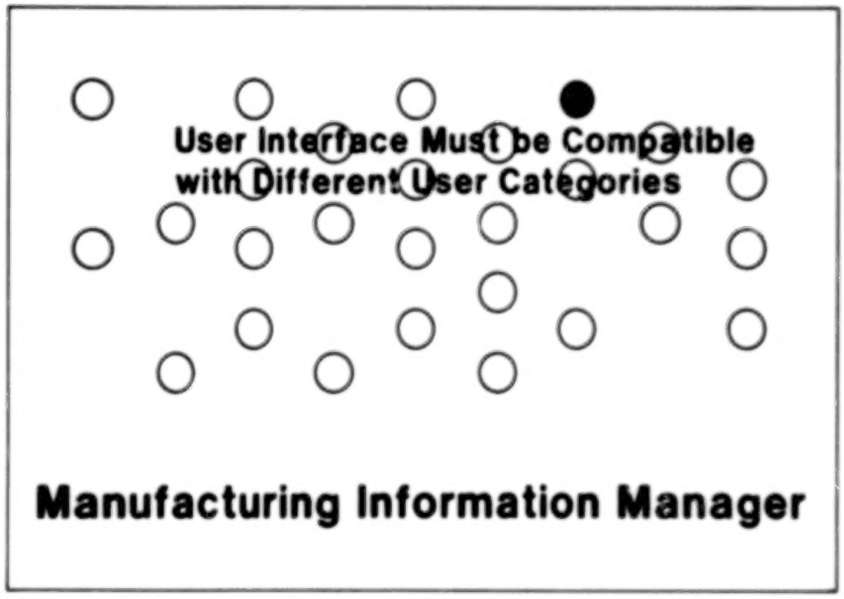
24. The MIM must support private data, that is, data from a local terminus that is not shared. This data, being nonreleasable, should remain optionally private. The MIM should allow this capability as an adjunct of local-global data management.



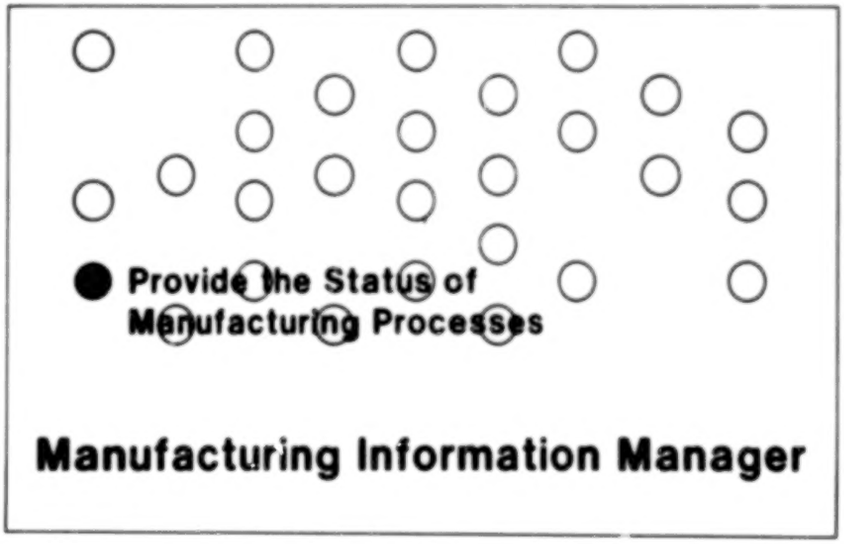
25. The MIM must provide a consistent, natural, and easy user interface. To most manufacturing personnel the user interface to the computer is the computer. The MIM user interface should be consistent between machines and independent of hardware and software configurations. The MIM user interface should have a format familiar to the user and should not demand any new skills for each category of user.



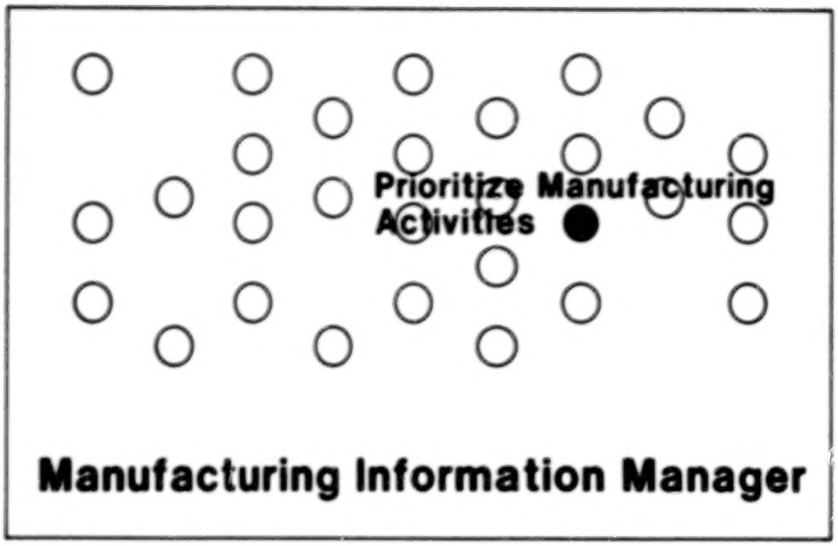
26. The MIM must provide a user language that is independent of external system languages and data models.



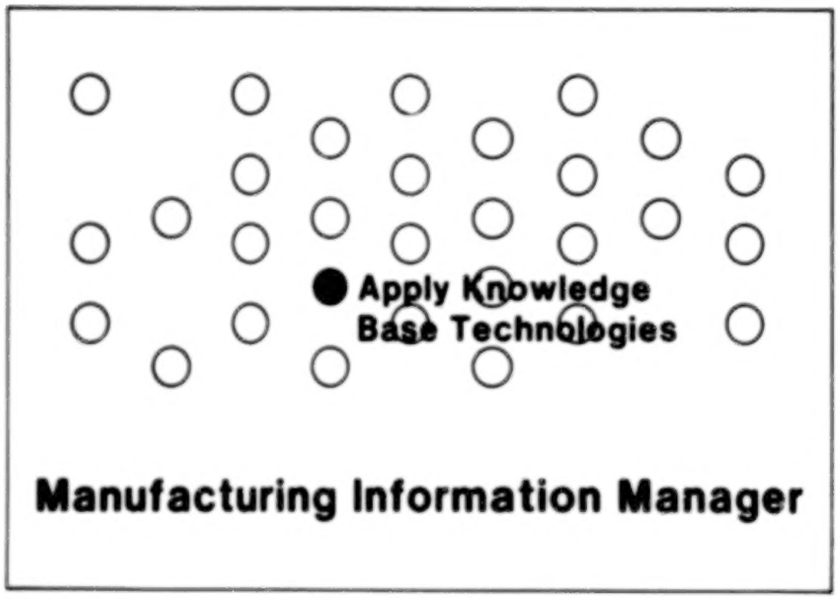
27. The MIM user interface must be compatible with different user categories, with a flexible language that meets the needs of each category of user.



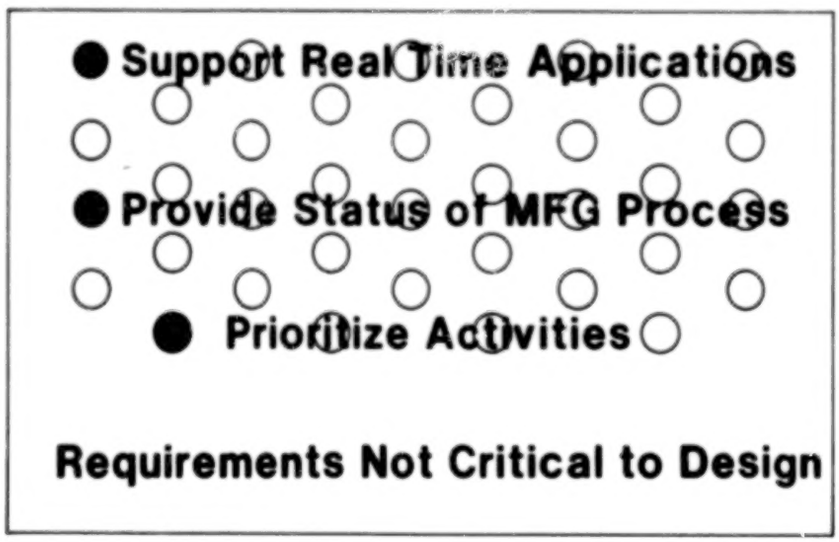
28. The MIM must provide the status of manufacturing processes and be able to report it at any given moment.



29. The MIM must prioritize manufacturing activities according to scheduling demands.



30. The MIM must apply knowledge base technologies and adapt automatically to changing manufacturing requirements, learning from experience by using artificial intelligence.

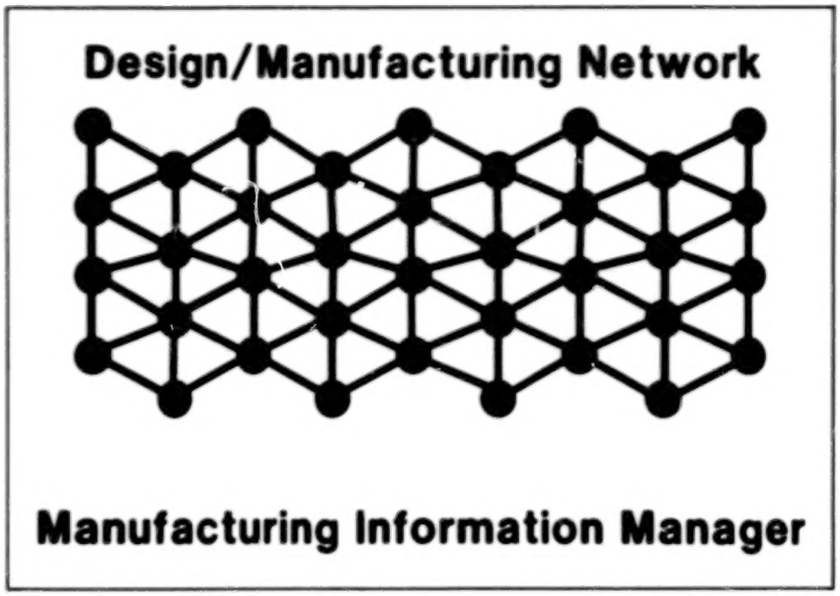


These manufacturing requirements also apply to design with the exception of the following, which do not:

The MIM must support real-time applications (7 above).

The MIM must provide status of manufacturing processes (28 above).

The MIM must prioritize activities (29 above).



Thus, 90% of the MIM requirements are shared by the design information manager (DIM) and the three exceptions (7, 28, and 29) would benefit DIM though they are not as critical to design as to manufacturing. It seems unlikely that design and manufacturing will ever be totally integrated or that design will accept the role of a support function of manufacturing. However, it is clear that a MIM which satisfies the thirty requirements enumerated here will satisfy the design information manager's needs as well. This fact will induce both processes to use an information manager as a mechanism that will result in the integration of design and manufacturing as a logical byproduct of the sharing and common management of data.

REFERENCES

1. O. L. Anderson and A. L. Calvery, "IPAD Requirements" (D6-IPAD-70040-D), Boeing Commercial Airplane Company, Seattle, December 1977.
2. Donald D. Meyer, "Reference Design Process" (D6-IPAD-70010-D), Boeing Commercial Airplane Company, Seattle, September 1977.
3. G. E. McKenna and D. D. Meyer, "Manufacturing Data Management Requirements" (D6-IPAD-70038-D), Boeing Commercial Airplane Company, Seattle, September 1980.
4. H. A. Crowell, "Product Manufacturing Interactions with the Design Process" (D6-IPAD-70011-D), Boeing Commercial Airplane Company, Seattle, June 1977.
5. D. L. Bernhardt, "Distributed Computing in the Engineering and Manufacturing Environment" (Task 3.10), Boeing Commercial Airplane Company, Seattle, June 1983.

A DISTRIBUTED DATA BASE MANAGEMENT FACILITY FOR THE CAD/CAM ENVIRONMENT

R. M. Balza
BOEING COMPUTER SERVICES COMPANY

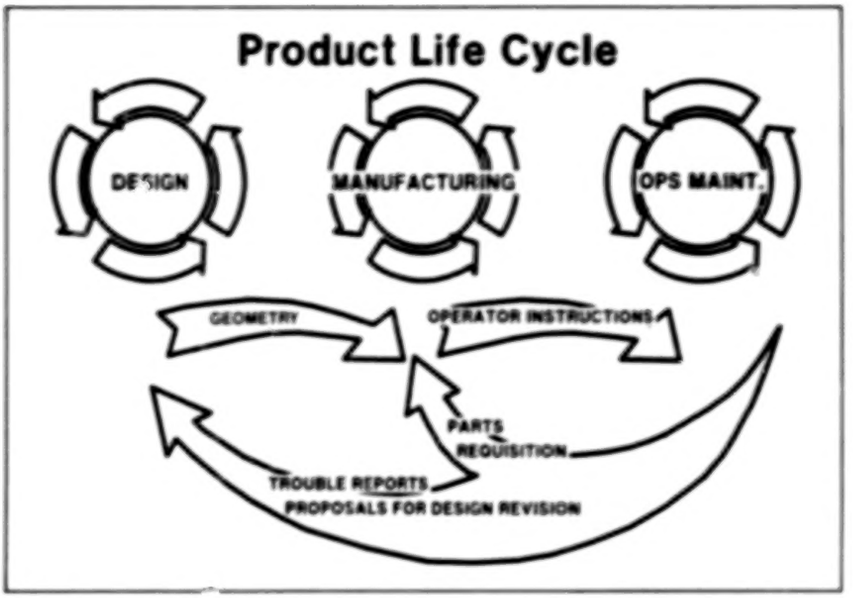
R. W. Beaudet
BOEING COMPUTER SERVICES COMPANY

H. R. Johnson
BOEING COMPUTER SERVICES COMPANY

ABSTRACT

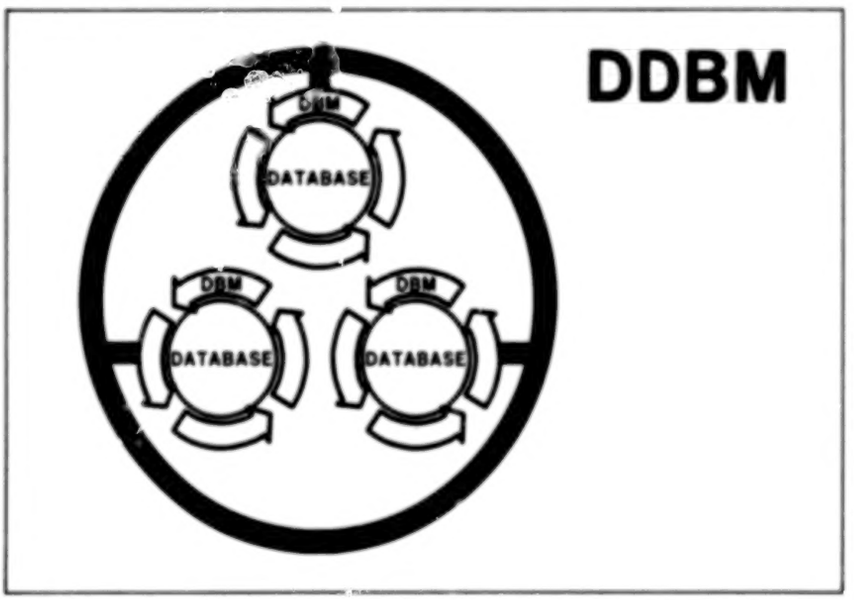
Current IPAD* research in the area of distributed data base management considers facilities for supporting CAD/CAM data management in a heterogeneous network of computers encompassing multiple data base managers supporting a variety of data models. These facilities include coordinated execution of multiple DBMSs to provide for administration of and access to data distributed across them.

*IPAD (Integrated Programs for Aerospace-Vehicle Design) development is performed by The Boeing Company under NASA Contract NAS1-17555. IPAD software and documentation may be obtained from the IPAD Program Management Office, The Boeing Company, P. O. Box 24346, Seattle WA 98124, M/S 73-03.

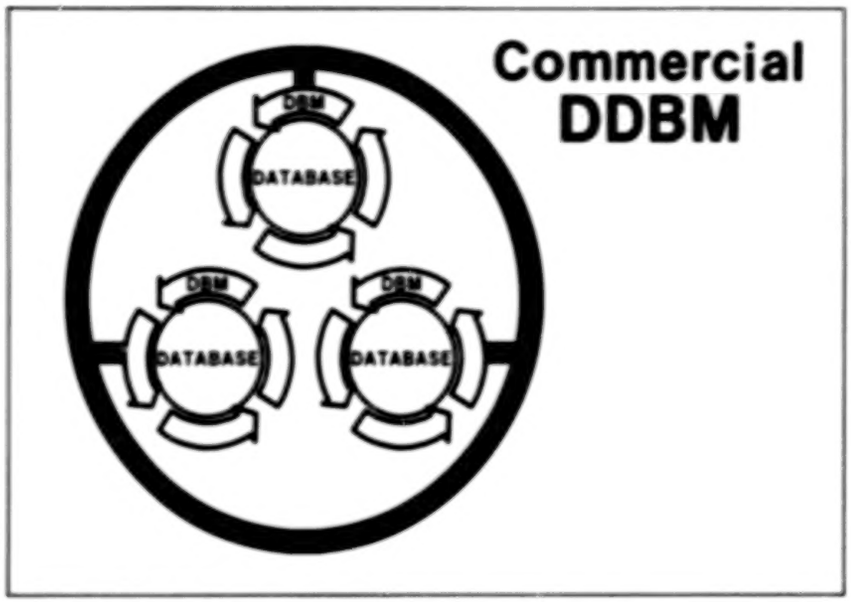


INTRODUCTION

Data base management (DBM) technology has evolved as a means of meeting some of the needs for managing complex, commercial file-oriented applications. In recent years, research has considered the application of DBMS technology to engineering environments. In particular, IPAD, a NASA/Navy-sponsored project, has considered applying data base management technology to all phases of the product life cycle: design (CAD), manufacturing (CAM), and operations/maintenance. IPAD has also investigated the application of DBM technology to the integration of processes within and between all phases of the life cycle through access to a common data base management facility.

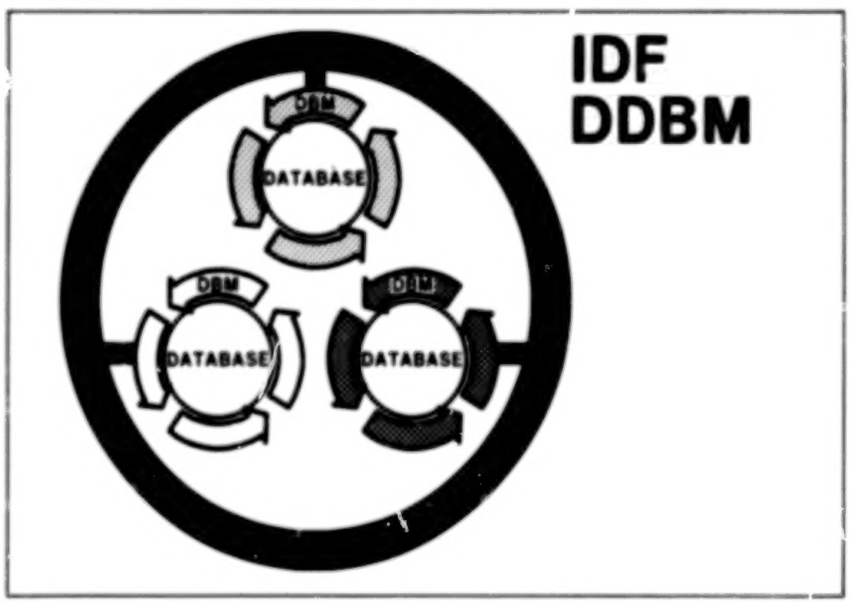


Whereas DBM technology evolved around centralized systems, much recent research has been devoted to distributed data base management (DDBM), which provides DBM facilities in a system encompassing multiple computers. Such systems offer opportunities for such benefits as reduced communication costs and increased availability of data. For example, a savings would result if a data set (data representing an engineering unit of work) were stored at the site where it is referenced ninety percent of the time rather than being stored at a central site. In keeping with sound practice regarding control of corporate resources, DDBM may provide for local autonomy over local data. Coexisting data bases may be tied together using DDBM. Thus data bases that have, for one reason or another, evolved at a number of sites may be accessed as if they were a single data base. DDBM may improve the availability of data because portions of a data base are likely to remain on-line despite the failure of some sites of the system. Applications available at selected sites of a distributed system may be applied to the entire data base, either by (a) accessing data remotely across the network or (b) providing local access by migrating data residence across sites of the system in much the same way that parts move between work stations in a manufacturing facility.



Some DDBM features are beginning to be incorporated into commercially available systems aimed primarily at business applications. The potential advantages of DDBM cited above are applicable to the engineering environment, which may span many sites supporting each phase in the life cycle of a product. Each of these phases may be active at any given time. Some of these sites, as in the case of a space station, may be extraterrestrial (ref. 1).

To date, commercial systems that do provide DDBM features encompass instances of a single data base management system (DBMS). Such a DDBM facility is said to be DBMS-homogeneous.

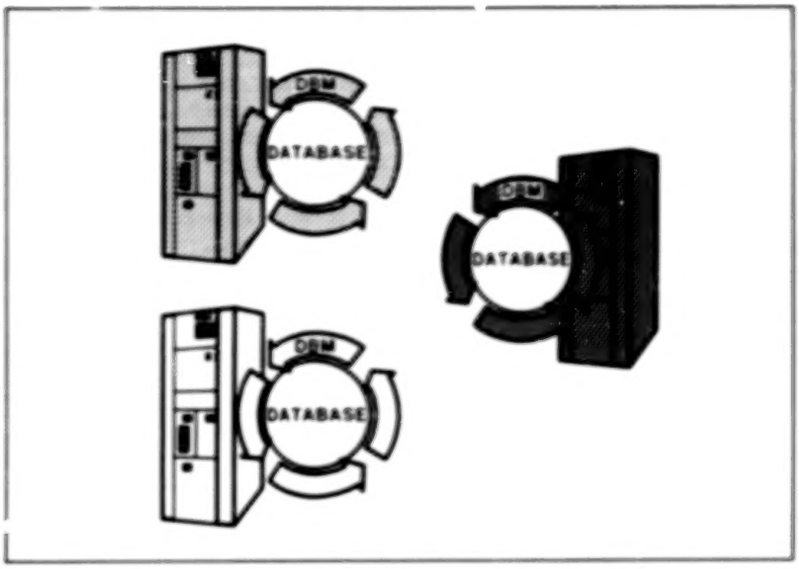


. . . It is not uncommon, however, for an engineering environment to encompass a heterogeneous mix of computers supporting multiple DBMSs and a variety of data models.

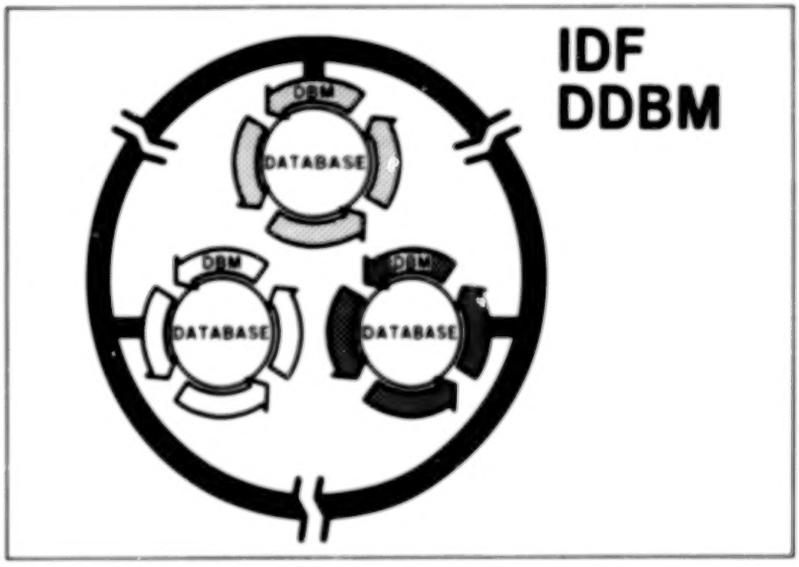
IPAD has launched a research effort to investigate heterogeneous DDBM facilities for such engineering environments. This paper describes the functionality of the IPAD distributed data base management facility (IDF).

OBJECTIVES

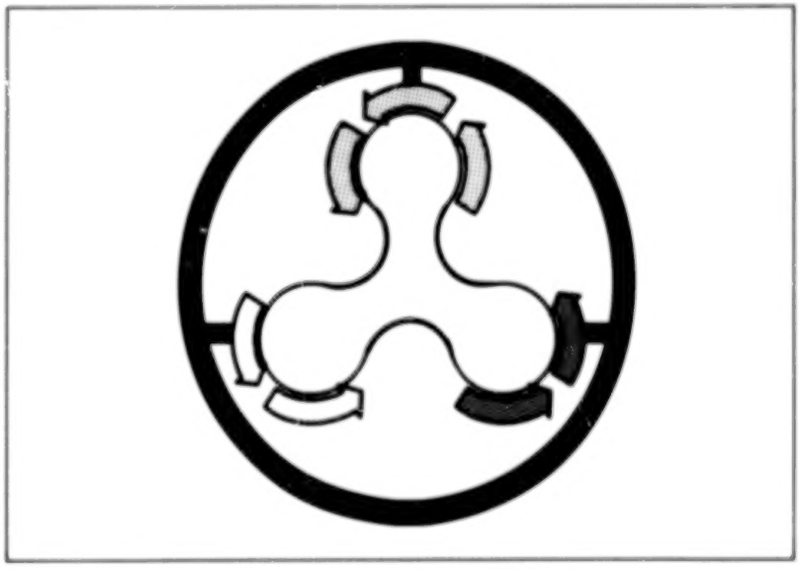
The functionality to be provided by the IDF is determined by several major objectives. These include:



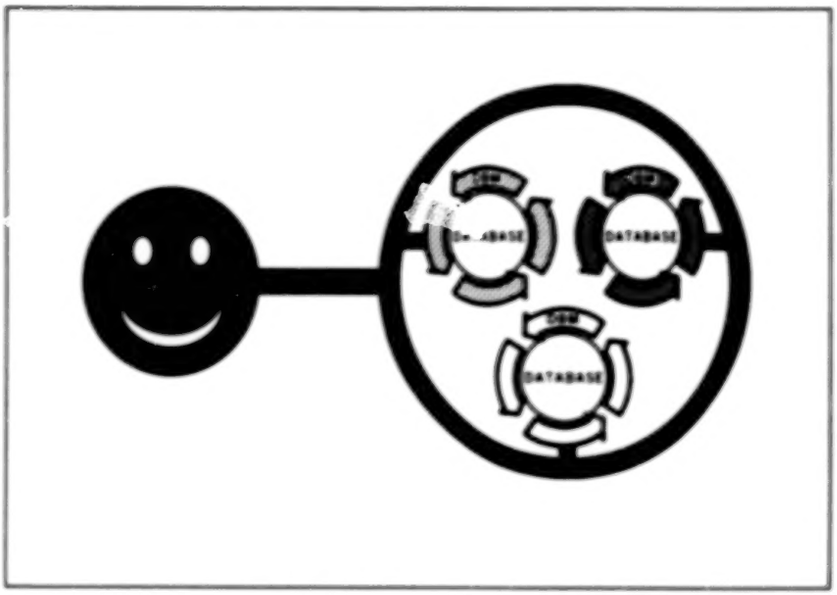
Heterogeneity of Hardware and Software. As discussed in the introduction, the typical product life cycle is supported by a heterogeneous mix of computers supporting multiple DBMSs and a variety of data models. A data management facility spanning these sites provides for integration of the applications that support the life cycle.



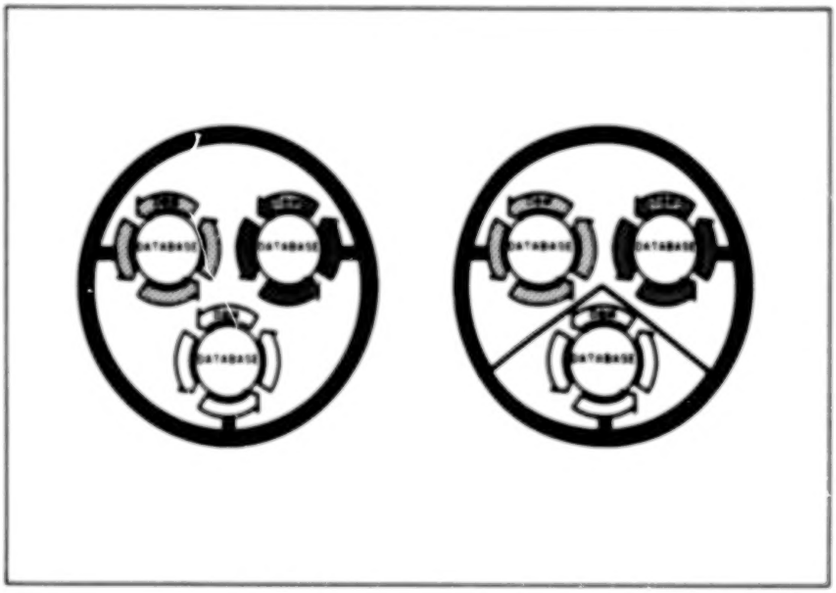
Site Autonomy. Services at a single node should be available to the greatest extent possible independent of other sites.



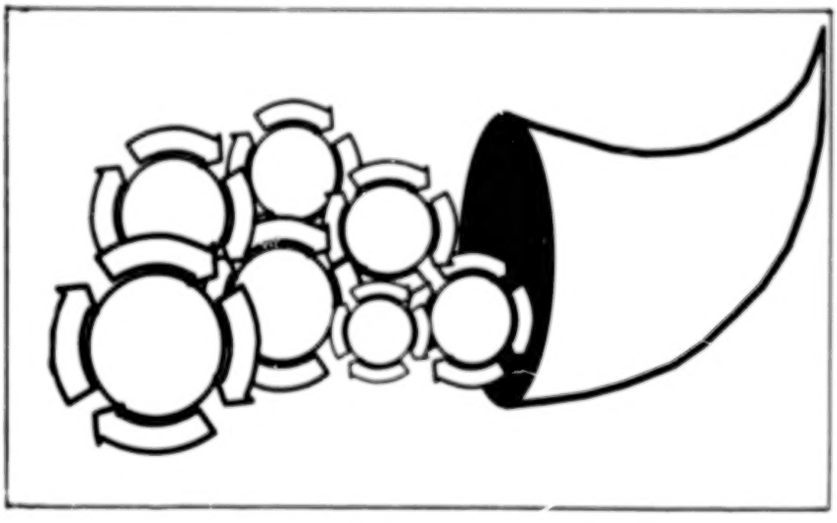
Transparency of Data Location. Users must be able to access data without having to know at which site(s) it resides. This is a convenience for the user, permitting data to be relocated or replicated without impacting the user or the user's programs.



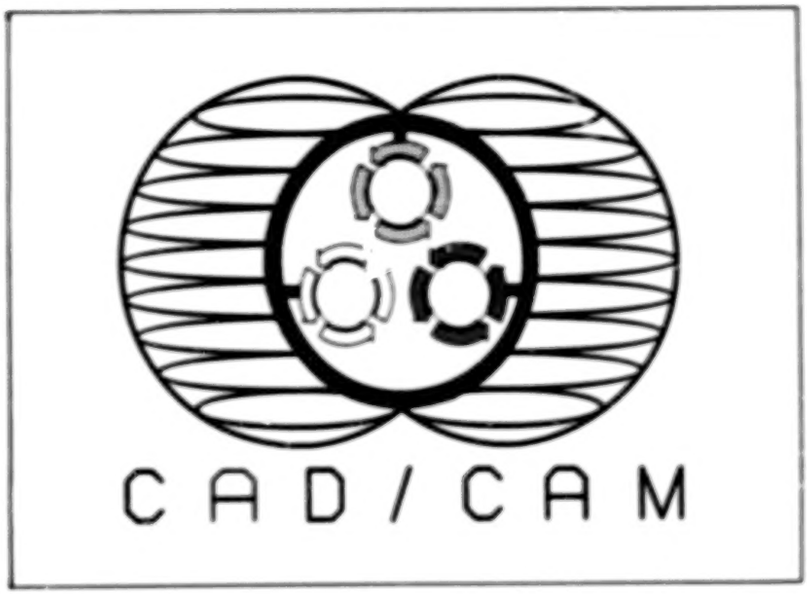
User-Friendly Interfaces. Users must be able to invoke IDF facilities in a convenient manner. Aspects of convenience include simplicity of use, familiarity, and uniformity. Simplicity involves easy-to-use interfaces, such as forms or menus, along with help facilities. Such facilities should accommodate novices as well as experienced users. Familiar interfaces are convenient because no additional training is required of users, and because these interfaces support existing applications without modification. Uniformity of interface allows users to work at multiple sites without having to deal with multiple data models. User friendliness, then, is relative to particular user/application mixes. The system should provide facilities to configure user interfaces from a common data model (DDL/DML), to be used at every site, to multiple data models, to be used at designated sites. It should be possible to tailor interfaces on an individual user basis.



Configurable Administration. To accommodate various organizational approaches to management, the system should provide facilities for configuring data administration from a centralized to a federated function. Configurability provides for various degrees of local administrative autonomy.



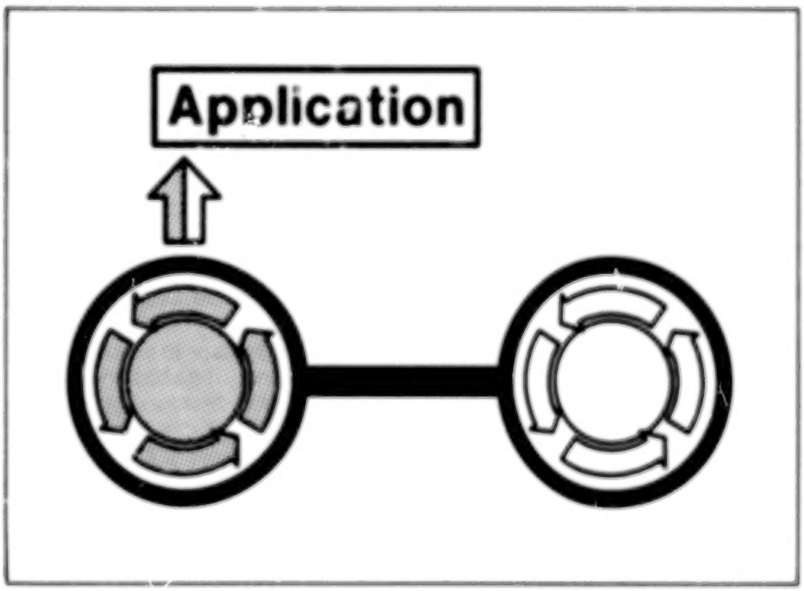
Open System Architecture. Many data models and DBMSs are available today. Initially, the IDF can incorporate only a limited subset of these. The system should provide an open interface to allow future inclusion of additional data base management systems and/or data models.



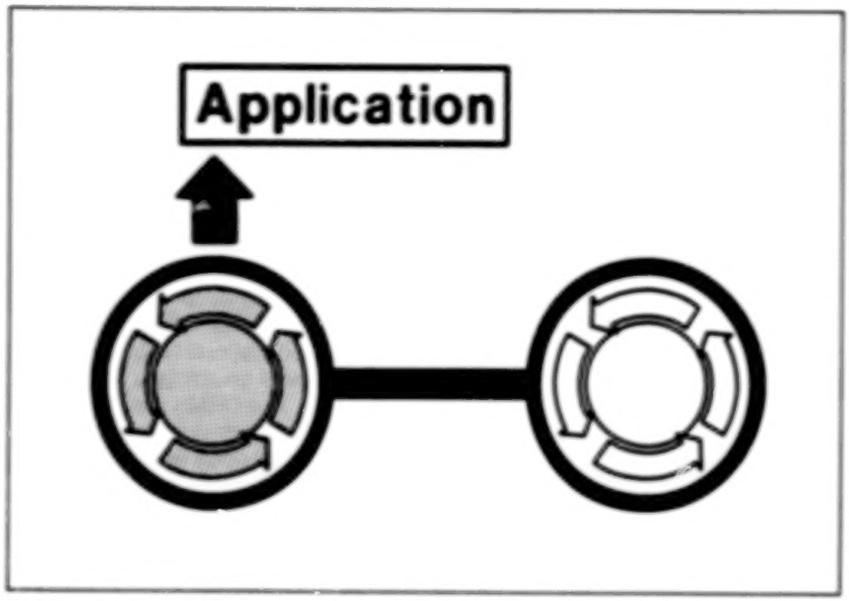
Traditional CAD\CAM Requirements. The system should provide facilities to support traditional engineering requirements such as scientific data types, geometry, data sets, versions, and configuration control.

OVERVIEW

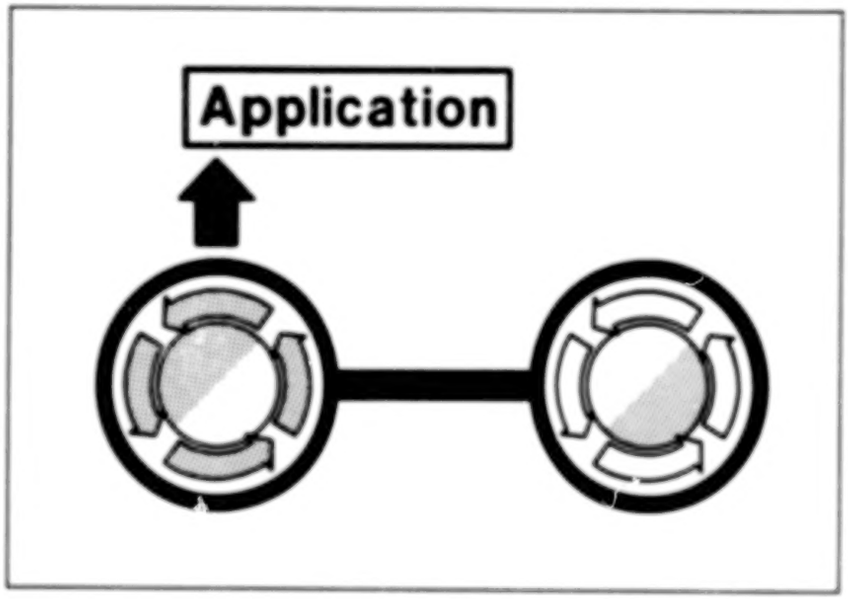
The IDF coordinates the execution of heterogeneous DBMSs residing on various network-connected sites to provide an integrated data base management facility. Conceptually, the IDF is implemented as a collection of software modules that execute concurrently on the various sites, communicating with local data managers and with each other across the network to achieve IDF functionality.



One batch or interactive command on a particular site can access data physically stored on one or more sites in the network, for example, issuing a retrieval command that obtains data from site 1 or site 2 or both, depending on the selection criteria in the command and the distribution of the subject data.

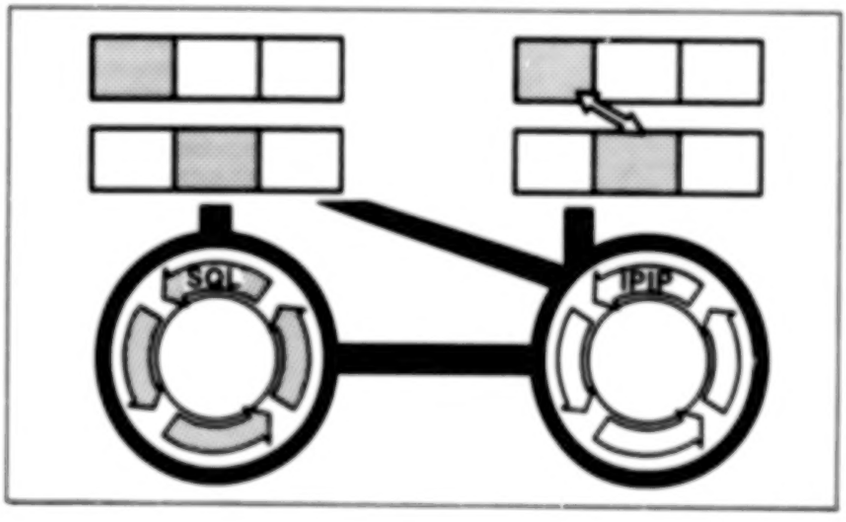


Data location is transparent to the user and consequently is not mentioned explicitly in the retrieval command.

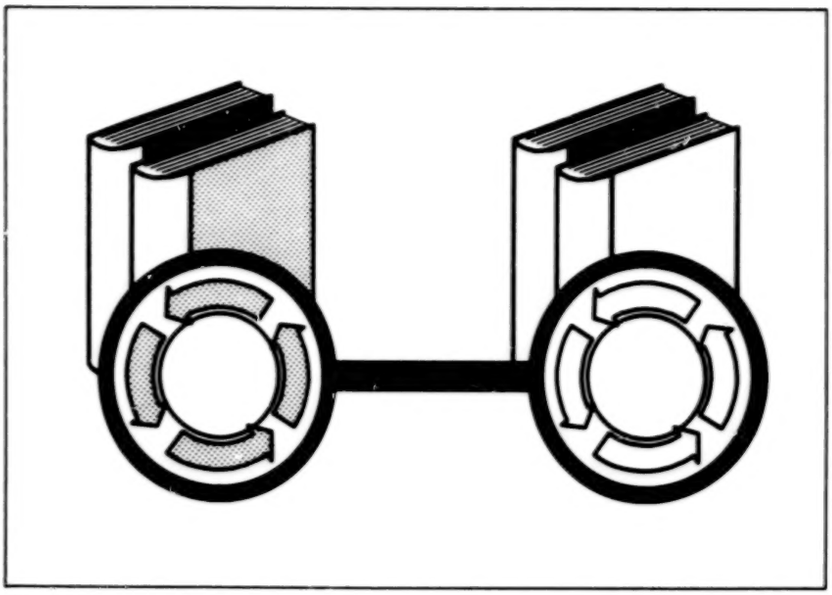


. . . Thus, data may be relocated within the network without affecting the user or the application.

Execution of a user command is driven by meta-data contained in the distribution data dictionary (DDD). In particular, the DDD contains directives regarding data distribution. This meta-data is used to determine site(s) from which data may be retrieved or site(s) to which data is to be assigned or reassigned. Thus, when an application stores data, it is assigned site residence according to meta-data in the DDD. Data may be relocated upon execution of an update command or various other events.

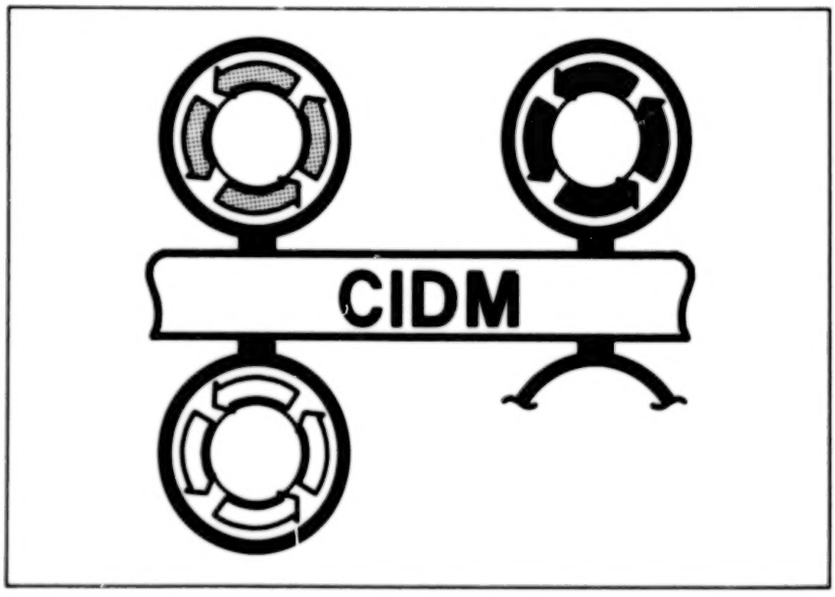


One or more data models (styles of describing and manipulating data) can be supported at each site according to administrator directive. Accordingly, a data model native to the DBMS at a particular site (e.g., the relational model supported by IBM's SQL at site 1, the network/relational models of IPAD's IPIP at site 2, or another data model supported by another vendor's data manager at site 3) might be used.



DISTRIBUTION DATA DICTIONARY

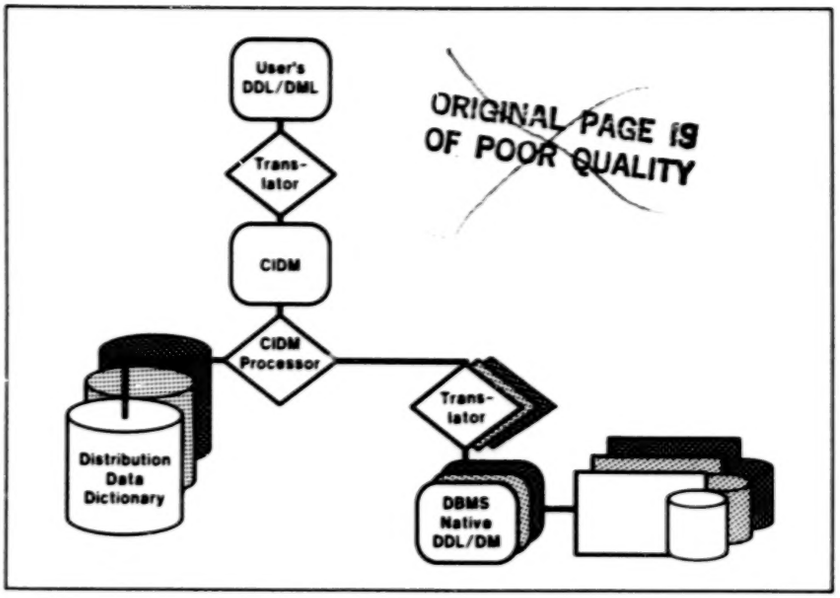
The distribution data dictionary (DDD) is an IDF-maintained repository of meta-data concerned primarily with the distributed aspects of the system (e.g., data location/relocation and facilities configuration). The IDF uses the DDD in conjunction with DBMS-native data dictionary facilities to coordinate the execution of DBMSs in the network. The DDD also contains meta-data required to compensate for semantic deficiencies of selected DBMS-native data models. For example, one DBMS might not support constraints on attribute values while others do. The DBMS may also contain meta-data about constraints that cannot be enforced at a single node (e.g., duplicates are not allowed for a relation that spans sites). For efficiency in implementation, the DDD may contain meta-data also residing in DBMS-native data dictionaries. The DDD is itself a distributed data base maintained by IDF. IDF processing (e.g., compilation, precompilation, and interpretation) of data description and data manipulation references and updates the DDD.



DATA MODEL TRANSLATION

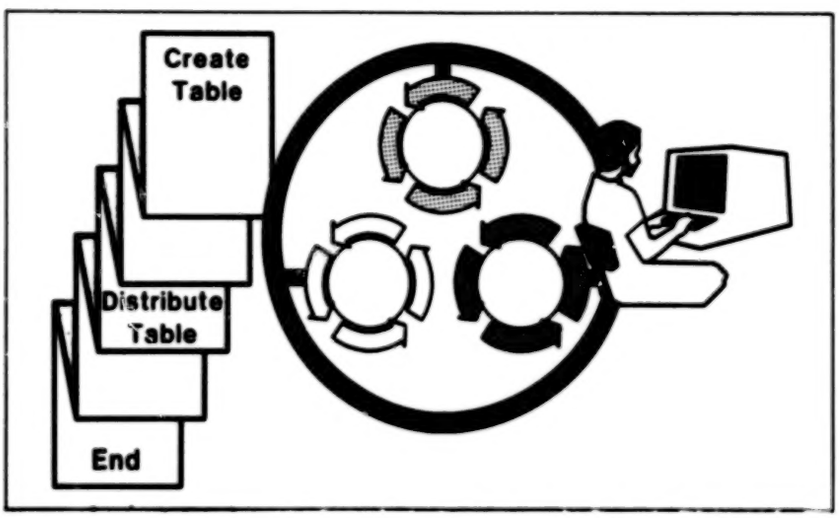
Many data models and DBMSs are available today, and many more are forthcoming. Initially, the IDF can accommodate at most two or three of these. However, the IDF provides an open system architecture to allow for future inclusion of additional DBMSs and/or data models. To this end, IDF supports a common interchange data model (CIDM) to which external (user-visible) data models are mapped (ref. 2). The CIDM provides for the declaration of entities, attributes of entities, relationships between entities, etc. The CIDM also provides for manipulation of data.

(-2



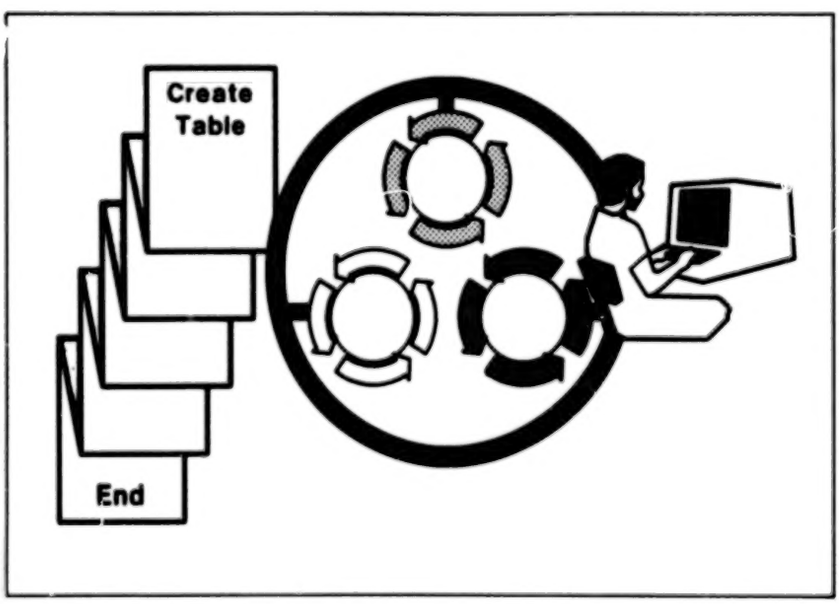
An external data model is incorporated into an IDF environment by writing software modules that translate data description and data manipulation statements of the external data model to/from their counterparts in CIDM. A statement in a user's data model is translated to its CIDM equivalent, which in turn is translated into the data model(s) native to one or more DBMSs in the network.

Data model translation may not be one-to-one. For example, a join operation issued by an application using the relational data model might be executed by an underlying DBMS supporting the network data model. In this case, the join is translated to CIDM and then into a routine containing several data manipulation commands for the network DBMS.



DATA DESCRIPTION

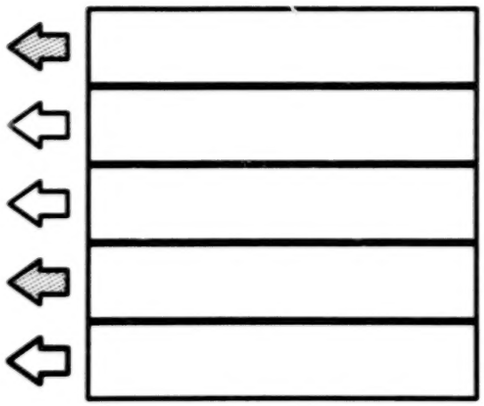
Data description facilities are provided by the IDF through translation to and from the CIDM, as discussed earlier. Data description capabilities provide for the description of entities, attributes of entities, relationships between entities, declaration of distribution policies, delegation of access privileges, and mappings between entities.



Distribution policies or rules are established by data administration either when a record/relation type is declared or at some time before actual instances of the data are stored. If no distribution rules are defined, the data will reside by default upon the site where it was declared.

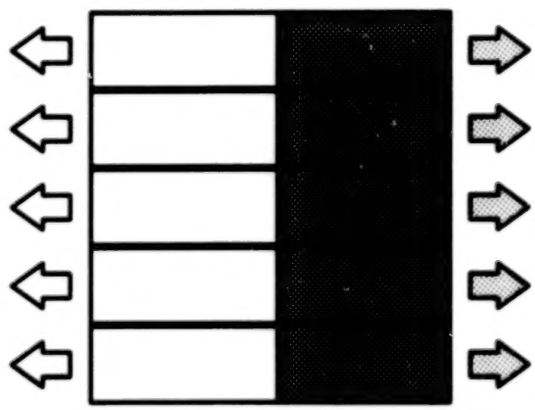
Data bases in existence before IDF software was installed may be registered with the IDF. In the absence of explicit distribution rules, data location defaults to the original site. In this case, existing non-IDF-compiled programs may access the data base. The data base may also be accessed from other sites, via IDF. Alternatively, the preexisting data base may be distributed at the time of registration with IDF. In this case, IDF processing of existing programs may be required.

HORIZONTAL DISTRIBUTION



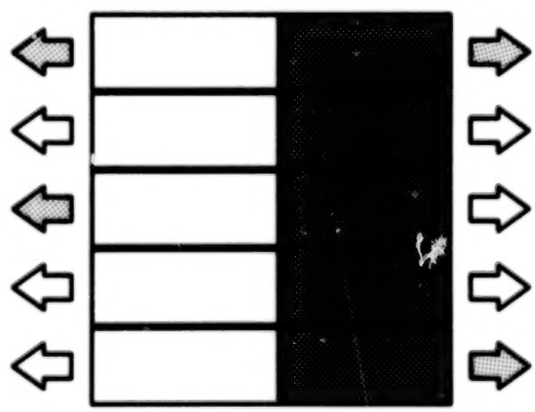
The IDF supports horizontal distribution, vertical distribution, and combinations of the two. Horizontal distribution distributes record instances according to various criteria such as frequency of use, predicates on attribute value(s), data set (data pertaining to a specific engineering unit of work), or data area (an arbitrary grouping of data sets). For example, aircraft part information may be horizontally distributed across sites by aircraft model number. Part information for Boeing 737s would be stored at a location in Renton, Washington, while part information for 747s would be stored in Everett, Washington.

VERTICAL DISTRIBUTION



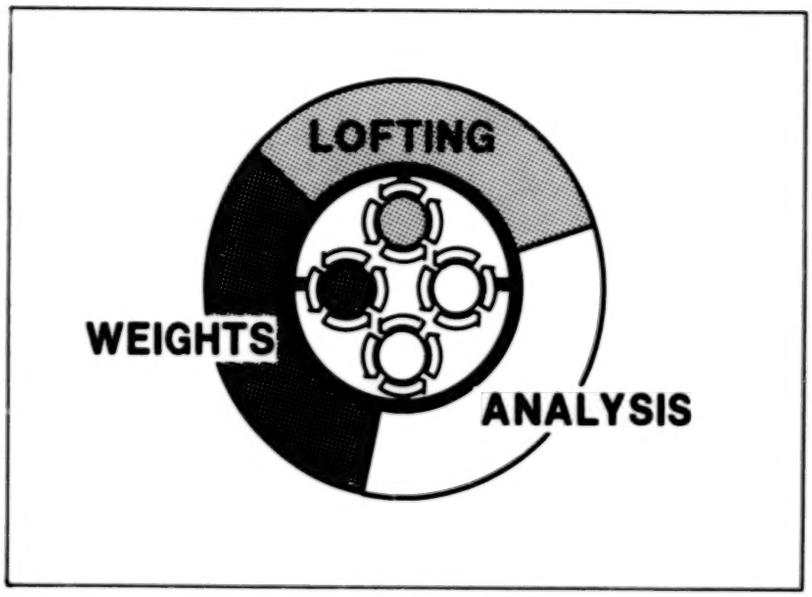
Vertical distribution decomposes a logical record instance into groups of attributes to be stored at different sites. A given site may retrieve a subassembly description record which is composed of technical data and cost information. Technical data may be stored at one site and cost data at another,

MIXED DISTRIBUTION

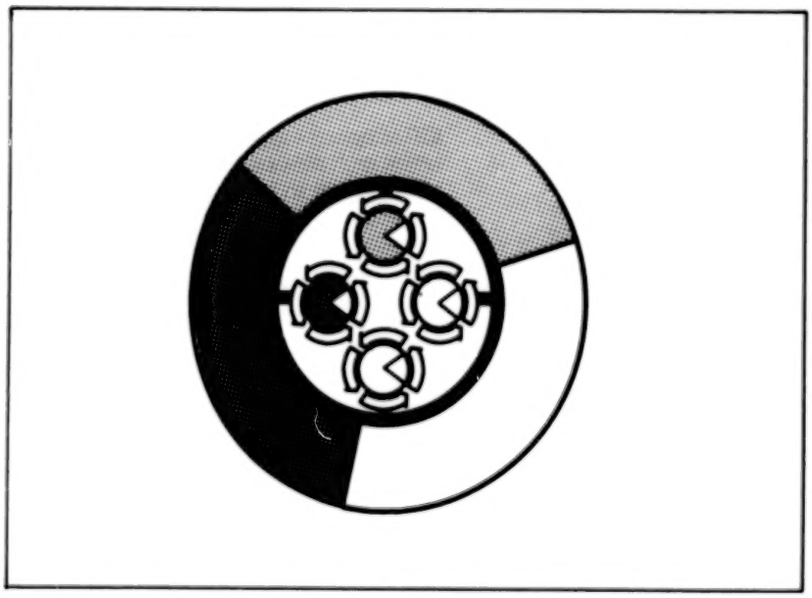


. . . or technical data may reside at the sites where the subassemblies were designed, while the cost data resides at the sites maintaining the subassembly inventories. In this instance, subassembly data is vertically distributed between design and inventory sites. At the same time, technical data is horizontally distributed among design sites and cost data is horizontally distributed among inventory sites.

Distribution policies also determine when and where data is to be replicated or migrated. Data communication cost is a major factor in the decision to replicate data. To minimize retrieval cost, data should reside on the site(s) where it is most frequently used. The decision to replicate, on the other hand, must take into account the cost of update to multiple copies.

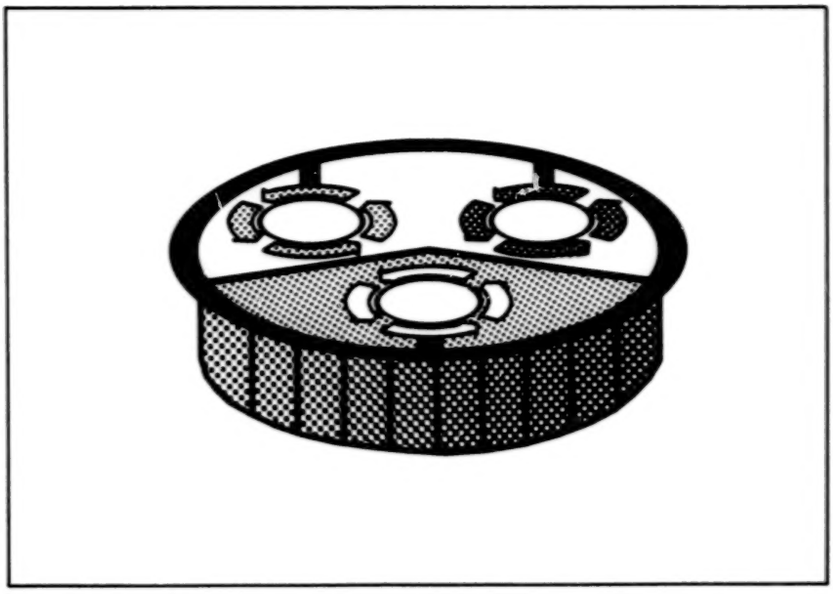


In the engineering environment, data supporting a unit of work is typically used concurrently by several task-oriented groups. An example in aerospace vehicle design is the release of a structural design for analysis, which must be performed by geographically separate groups such as lofting, weights, and structural analysis.

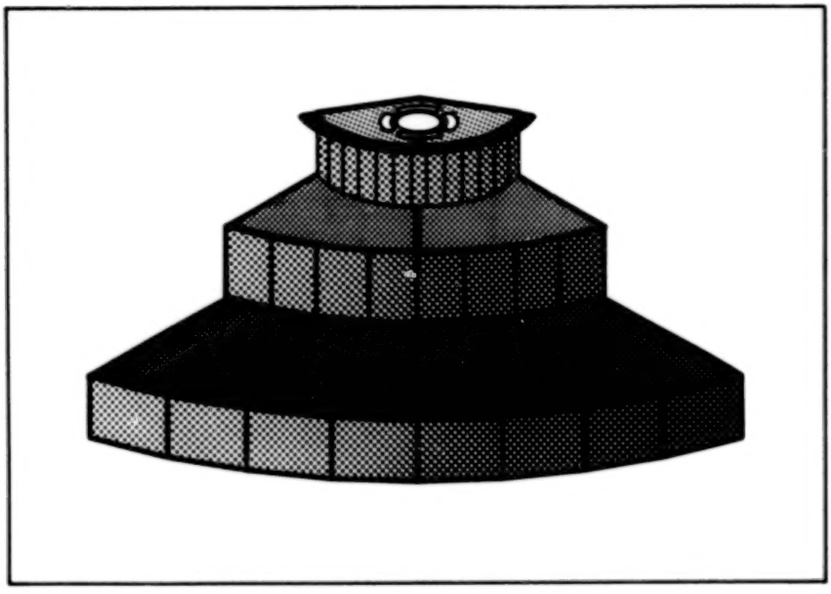


. . . The system monitors the frequency of access to the structural design information. Based on the declared distribution policy, the structural design data would be migrated or replicated when remote usage exceeds a specified retrieval/update ratio. This, in turn, would make the majority of data access local, thereby decreasing communication costs.

Through the use of distribution data dictionary and specified distribution policy, frequency-of-use information is maintained. This information is available for both the automatic relocation of data and for use by the data administrator in tuning the system.



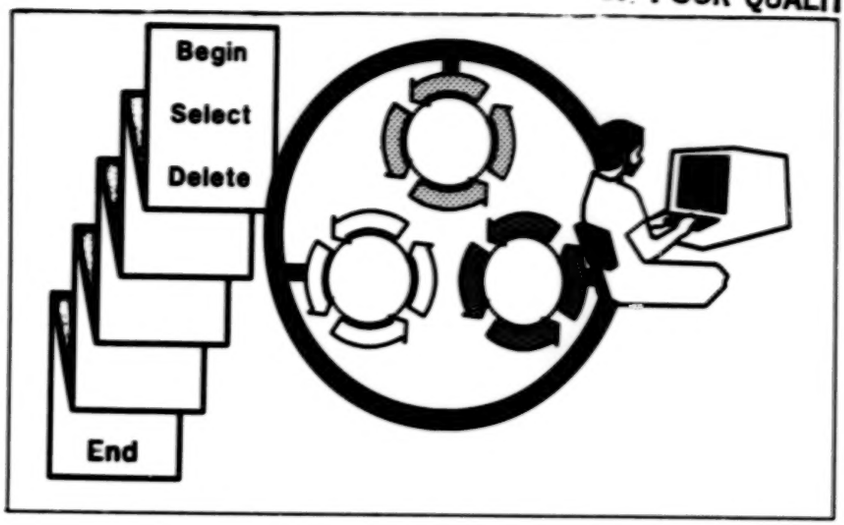
The IDF provides mechanisms for delegation of access priviledges for data description and manipulation, including permission to (1) read data, (2) update/delete data, (3) read data descriptions, (4) update/delete data descriptions, (5) utilize existing data descriptions in creating views, and (6) delegate granted privileges.



The delegation mechanism is used to configure the data administration function for an enterprise to (1) be the responsibilty of a single, centralized team of administrators or (2) assign responsibilty to a team of administrators at each site or group of sites. In the latter case, authority may be delegated to these teams in a way that parallels corporate structure. Administration is said to be federated in that each team is given a degree of autonomy but must work in concert with other teams to make data available across the network.

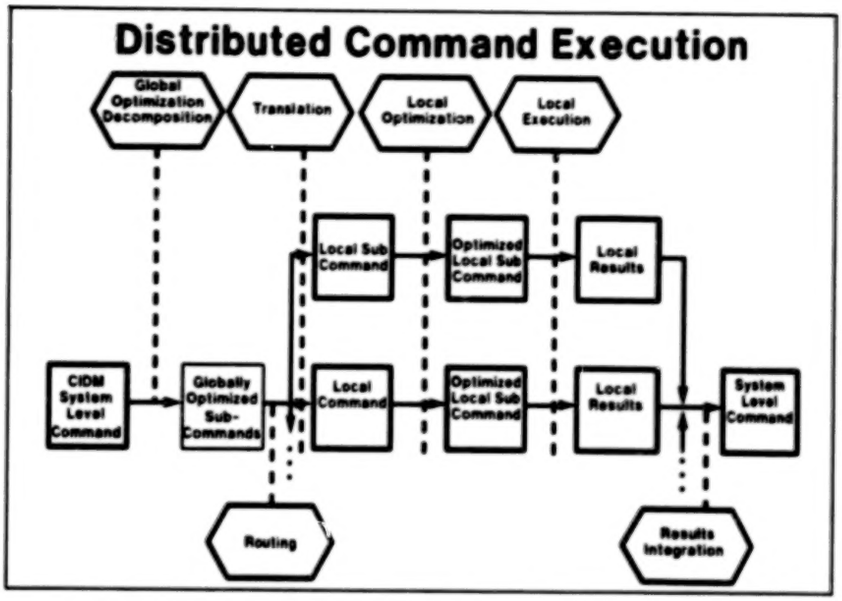
IDF supports mapping functions through which multiple existing data bases may be mapped into an integrated view or views. For example, one could create the join of a relation (declared in a relational data base) and a record (declared in a network data base). The result might be mapped to an integrated view in the relational or network model.

The IDF reports intersite conflicts such as the same name appearing for different entities or different field widths for the same attribute. Administrative tools are provided for resolving such conflicts.



DATA MANIPULATION

Data manipulation facilities are provided by the IDF through translation to/from the CIDM, as previously discussed, and furnish location-transparent access to data bases distributed across the network.

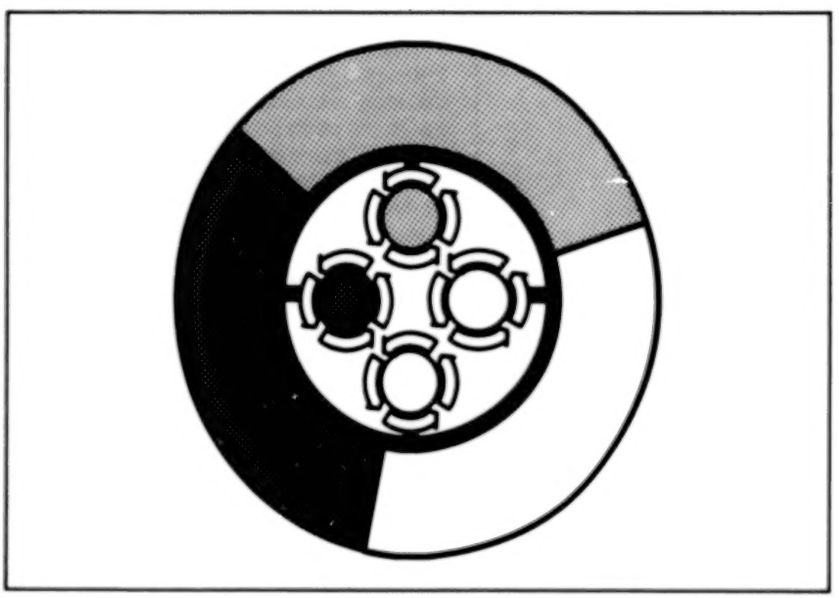


System-level commands enter IDF in either a DBMS-native format, such as that of IBM's SQL, or some other format such as that of the entity-attribute-relationship data model. In any case, the commands are translated to CIDM format and decomposed into local subcommands to minimize network traffic and gain sorting and merging efficiency. These subcommands are routed to their target site processor, where they are translated to DBMS-native format. The translated subcommands are submitted to local data managers such as IPIP (refs. 3,4), RIM (ref. 5), or SQL (ref. 6), where they may be optimized with respect to input/output overhead. Local results are collected by the target site IDF and shipped back to the originating IDF for integration and presentation to the requesting terminal or program.

System-level commands may be entered via interactive or programming language interfaces. Interactive interfaces provide query and update capabilities to both the novice and expert user. These interfaces provide full retrieval, update, and navigational services; total (tally) and computed value materialization functions; and complete display and terminal interface capabilities (graphics, function support, report generation, panel display, and definition). Cost limit supervision on long-running queries is provided to inform users of query cost and resource limitations.

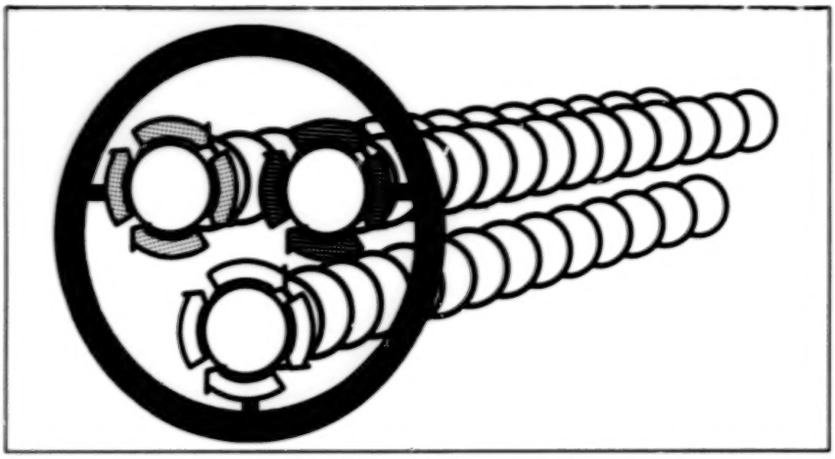
Programming language interfaces are supported by precompilation and binding facilities. Programming tools are provided to aid in the declaration of static work areas. Additionally, IDF provides for security and transaction management. A transaction—a unit of work that takes the data base from one consistent state to another—may span a single data manipulation command or

aggregates of commands. The IDF utilizes locking and commit mechanisms to ensure data integrity (ref. 7). Before updates to data are committed, the system ensures that all involved sites are ready to commit. If one or more of the involved sites are not ready to commit, the transaction is backed out (ref. 7).

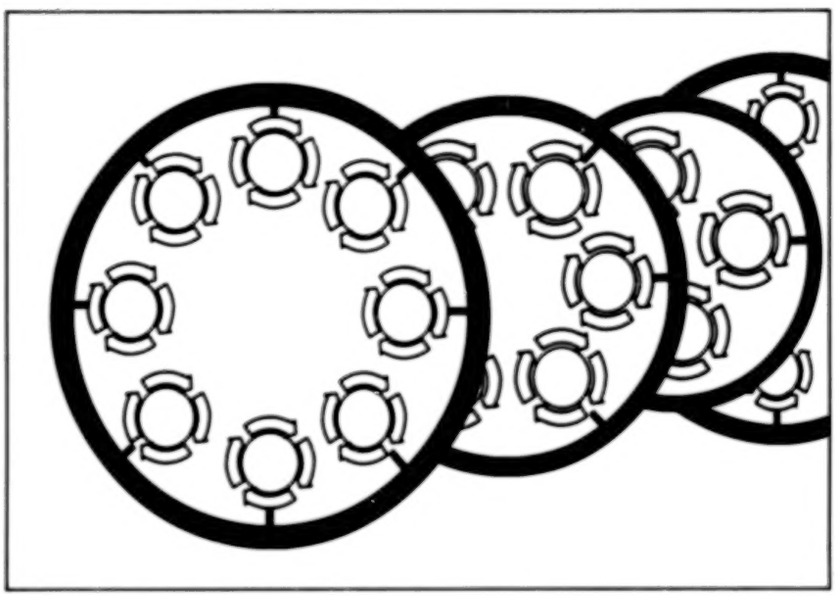


Updates to replicated data are synchronized by the IDF. Synchronization is not necessarily performed in real time but within a time span appropriate for engineering requirements.

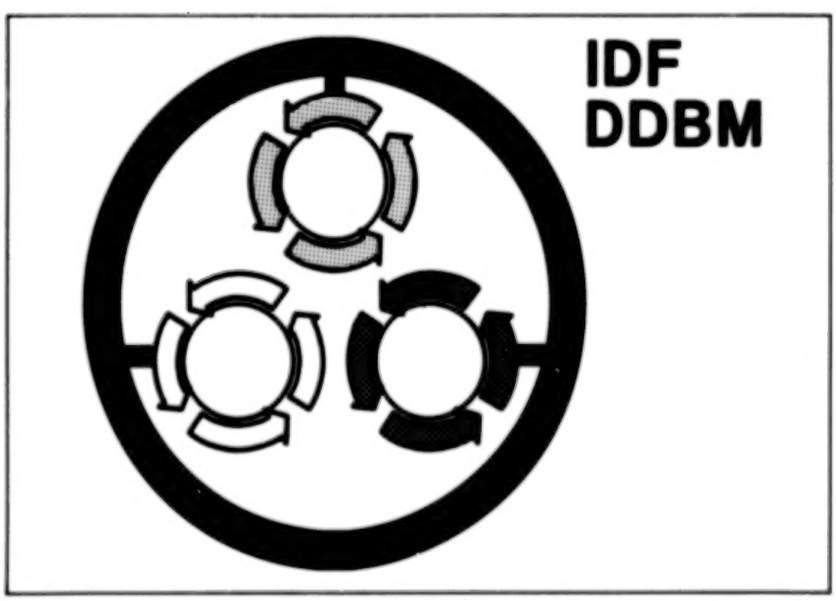
The IDF provides various functions for system monitoring and tuning (e.g., specifying local and global parameters for audit purposes, support for the configuration control of data, and the capture of update/retrieval statistics).



BACKUP AND RECOVERY
Full backup and recovery facilities are provided by the IDF through the use of transaction logging mechanisms and commit protocols. These include rollback and rollforward capabilities. The IDF utilizes and coordinates the recovery facilities of the individual DBMSs within the network.



FACILITY CONFIGURATION
Services are provided for configuring an IDF facility. These include the addition or deletion of sites, the addition or deletion of individual data base management systems, and the specification of communication paths between IDF sites.



SUMMARY
IPAD project activities described in this paper are in the formative stage of what is expected to be a three-year effort. Functional specifications of IDF and limited prototyping are planned for fiscal year 1984.

REFERENCES

1. "Computer-Assisted Engineering Data Base — Distributed DBMS Support for the Space Station Life Cycle," R. P. Dube and H. R. Johnson, The Society of Mechanical Engineers, 83-WA/AERO11, November 1983.
2. "Network and Relational Modelling in a Common Data Base Architecture Environment," H. R. Johnson, J. A. Larson, and J. D. Lawrence, Sperry Univac Research Report TMA00720, March 1979.
3. "Data Base Administration User Guide, IPAD Information Processor (IPIP) (Version 5.0), CYBER/VAX Configuration," IPAD document UM-REL5-200, The Boeing Company, Seattle, December 1983.
4. "Application Programming User Guide: Interfacing and Integrating Application Programs (Version 5.0, CYBER/VAX Configuration)," IPAD document UM-REL5-300, The Boeing Company, Seattle, December 1983.
5. "BCS RIM — Relational Information Management System User Guide," version 6.0, RIM document 70101-03-017, July 1983.
6. "SQL/Data System User Guide," release 2, IBM configuration, IBM document GH24-5042-0, April 1983.
7. Notes on Data Base Operating Systems," *Advanced Course on Operating Systems*, J. N. Gray, Technical University of Munich, Elsevier North-Holland, New York, 1977.

DATA MANAGEMENT FOR COMPUTER-AIDED ENGINEERING (CAE)

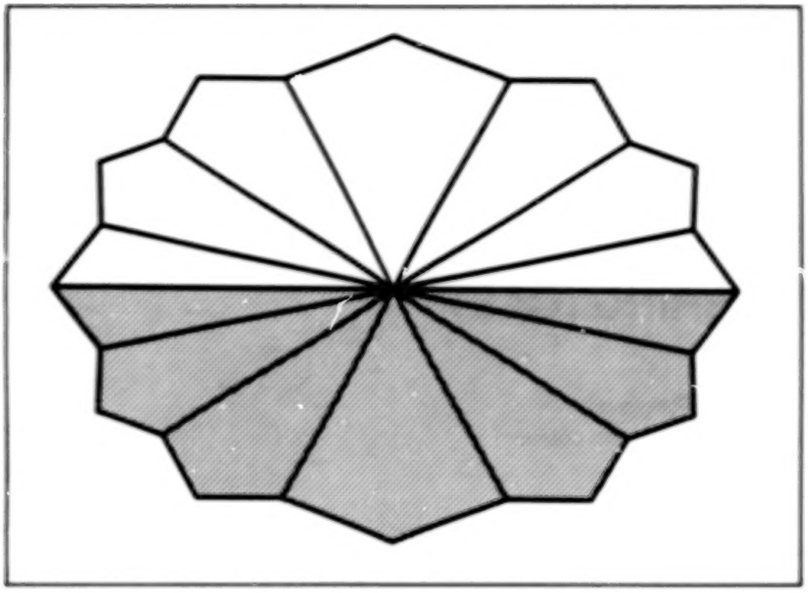
W. A. Bryant
BOEING COMPUTER SERVICES COMPANY

M. R. Smith
BOEING COMPUTER SERVICES COMPANY

ABSTRACT

Analysis of data flow through the design and manufacturing processes has established specific information management requirements and identified unique problems. The application of data management technology to the engineering/manufacturing environment addresses these problems. An overview of the IPAD* prototype data base management system, representing a partial solution to these problems, is presented here.

*IPAD (Integrated Programs for Aerospace-Vehicle Design) development is performed by The Boeing Company under NASA Contract NAS1-17555. IPAD software and documentation may be obtained from the IPAD Program Management Office, The Boeing Company, P. O. Box 24346, Seattle WA 98124, M/S 73-03.

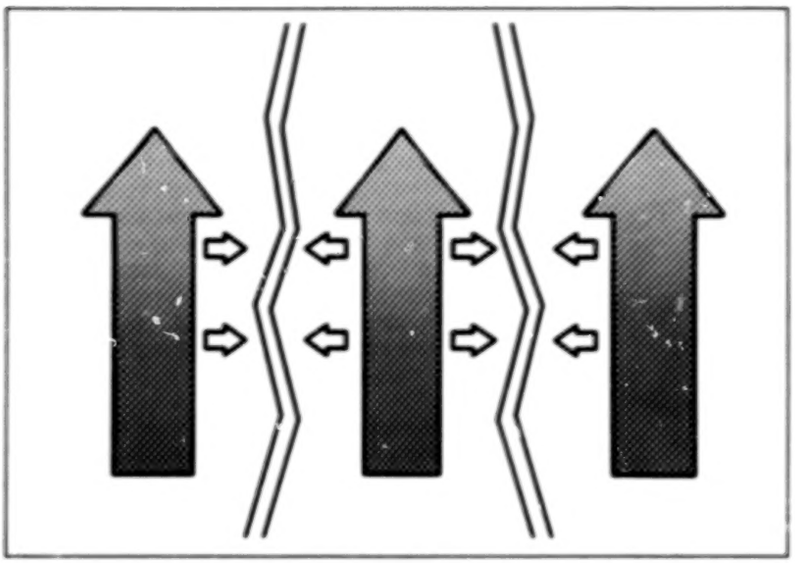


INTRODUCTION

The technological advances that have permeated the engineering/manufacturing environment in recent years have greatly reduced the time required to manufacture products, but they have also created "islands of productivity"—functions isolated, to the detriment of the overall process, from other functions with which they should interrelate. Data flowing through the pipeline must force its way through many bottlenecks along the way. As further advances are achieved in areas such as engineering work stations, robotics, and automated manufacturing, the problems inherent in transferring data from point to point increase. What has caused these problems, and what solutions appear on the horizon?

To find the answer to these questions, one must first examine the engineering and manufacturing processes through which the data flows. The IPAD project, currently co-sponsored by NASA and the Navy, has devoted years of research to this subject and has documented extensively the design and manufacturing processes as they are normally conducted within the aerospace industry (refs. 1,2,3). Despite the aerospace orientation of this research, the lessons learned can be applied in the wider industrial sphere as well. This paper will address some problems and solutions in today's data management arena and some steps being proposed for the future.

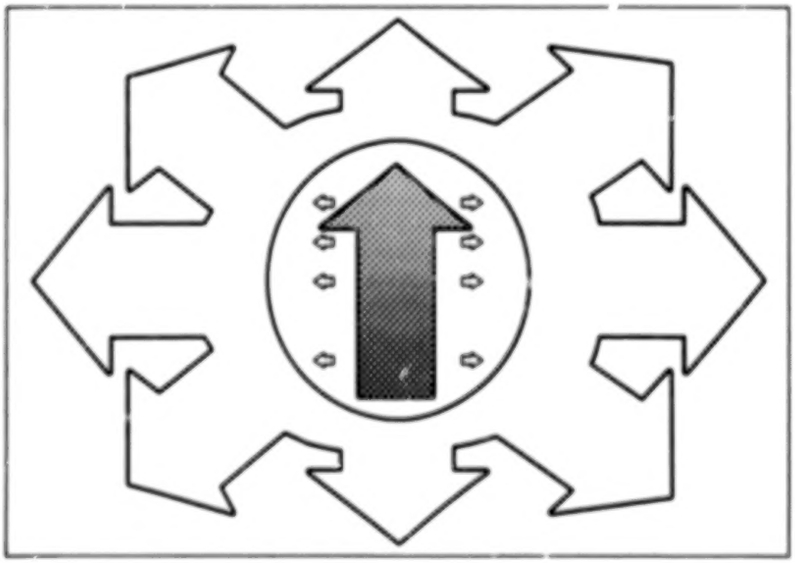
IPAD studies (refs. 4,5,6,7) conclude that the most promising solutions to problems of productivity within a typical large company lie not in the automation of shop operations or the computerization of design—worthwhile as these innovations are—but in managing and controlling the company's information base. IPAD has produced a prototype data base management system (DBMS) as a partial solution to these problems. This paper will describe the design of this software, its current capabilities, and some areas of further study now in progress. It is our firm belief that this technology will greatly assist industry in its search for better ways to manage information and improve productivity.



THE FACTS OF THE ENVIRONMENT

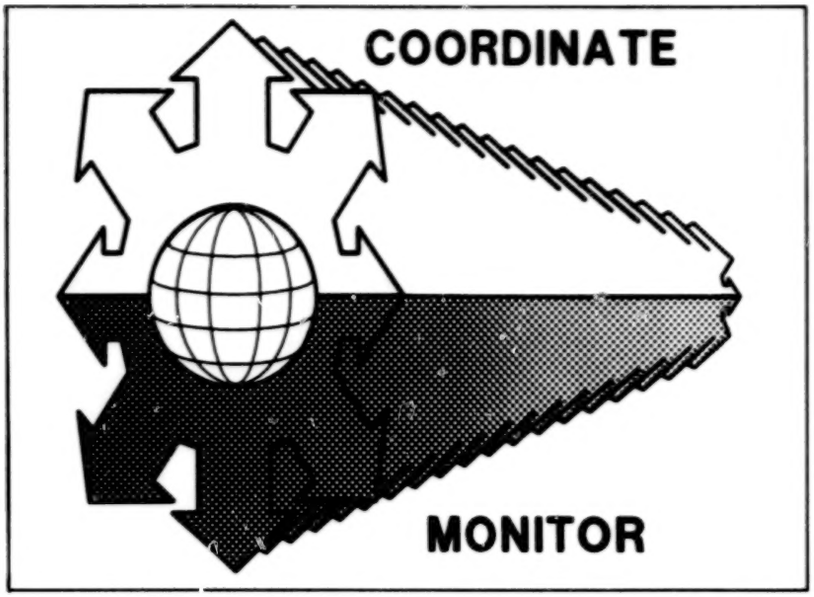
When we look at the engineering/manufacturing environment today, we see two major characteristics that emerge as problems: (1) The environment is fractured, and (2) a high degree of vertical productivity exists within the various disciplines of a company but horizontal productivity is somewhat lacking.

A "fractured" engineering/manufacturing environment exists where the design engineer and manufacturing engineer are bombarded by inputs from all over the company through various means (computers, other people, existing documentation) and in various forms (electronic presentation, mail services, and conventional drawings). In addition, high-technology systems are constantly evolving and, as they come on the market, are being incorporated into industry, further complicating information management. It is the monumental problem of industry today to gather, coordinate, correlate, integrate, and validate information from all of these sources in order to design, engineer, and manufacture a product.



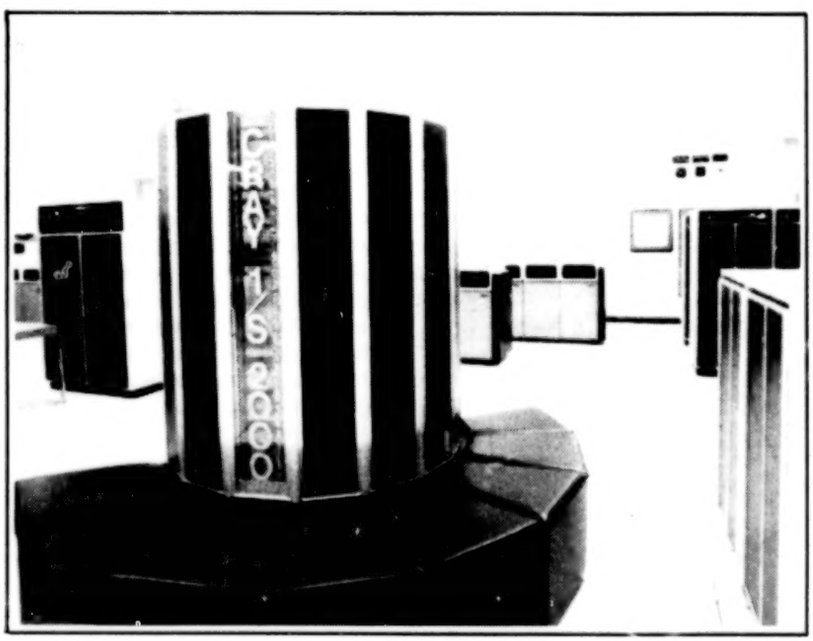
Today companies face a problem that will have an increasing impact as computer technology expands: the lack of horizontal communication between various disciplines. Each discipline maintains a high degree of productivity, which we will call vertical productivity, through the use of very sophisticated application programs. In many instances, executive software is employed to manage a discipline's various applications required to solve unique problems. This same degree of productivity is not achieved between disciplines.

Data required to support the engineering-manufacturing environment is often replicated across the various disciplines and as a result becomes very difficult to maintain as a consistent set or version throughout the whole process, as indicated in the following scenario:



The engineer/designer produces a design through various computer programs in the form of design information listings and design drawings, which are then used by detail designers to produce or create drawings and data required by manufacturing. This may be accomplished through turnkey graphics systems or by analysis of detail design drawings and then by examining the stock of material available; deciding how to produce the product; and programming the numerical control systems, robots, and material movers that exist in today's manufacturing environment.

But what if this process should be interrupted by additional design criteria or other engineering/manufacturing change requirements? Often data must be reentered into the process. Someone now must (1) coordinate the change process to ensure that all required redesign and analysis steps are taken to ensure the quality of this end product and (2) monitor the total information base, inspecting the product to ensure that the integrity of all the data is maintained at all times.

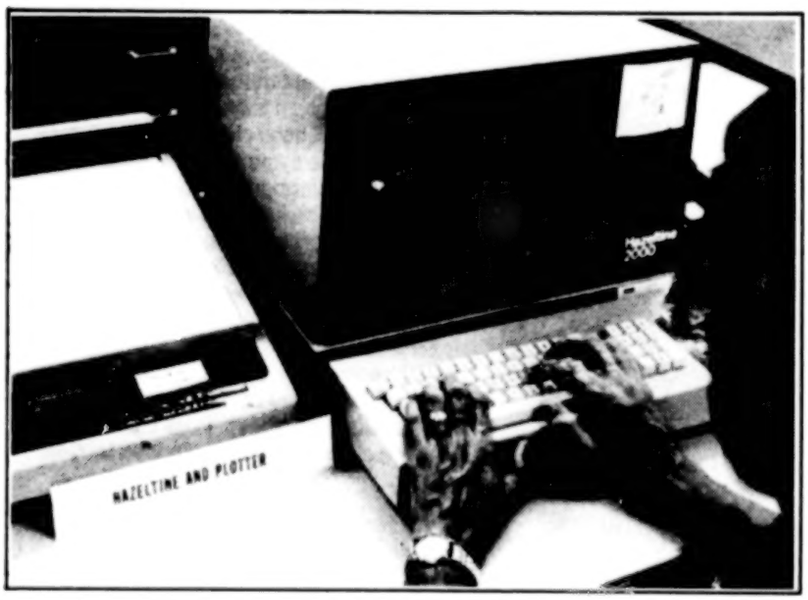


Because of the information processing requirements that such changes impose, companies have responded by placing arrays of highly sophisticated computing hardware at strategic locations throughout their operations. In many cases, each of these was chosen to meet a particular need or to satisfy some special requirement. Many small islands of computing power, each serving the need of a single location, have developed. Often a data base management system (DBMS) or file management system exists at these isolated nodes to manage various types of data (project data, geometric data, etc.)

From this it should be apparent that today's typical engineering/manufacturing environment includes many different computers as well as a variety of data management systems. These hardware and software systems need to be able to communicate with each other in order to facilitate the flow of information.

THE DIRECTION INDUSTRY IS TAKING TODAY TO SOLVE THE PROBLEM

Some companies have tied many of these nodes together into a network of computers. However, in order to effectively communicate, a communication protocol must be established and software written to provide for the various communication functions desired. Problems associated with data translations between the various machines also arise. Differing word sizes, diverse internal data representations, and varied applications and devices performing similar tasks with dissimilar representations of data—these complexities make the transfer of information from machine to machine an extremely challenging undertaking.

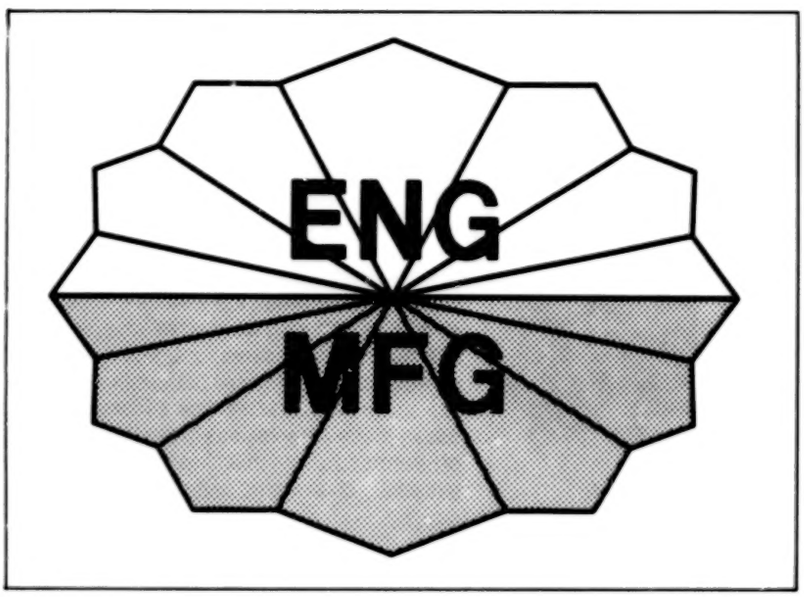


Many application programs are usually available to help the engineer/designer accomplish a task. Formerly, this individual would have been responsible for entering the data and job control language to execute the programs and would have had to know the user interface of every program used and, in some cases, understand the job control language of other computers and operating systems as well. In an attempt to integrate these applications to provide a more productive environment for the engineer, most companies now provide a single-user interface to the designer, while the management of the actual flow of data between these programs and the program execution are handled by executive software.

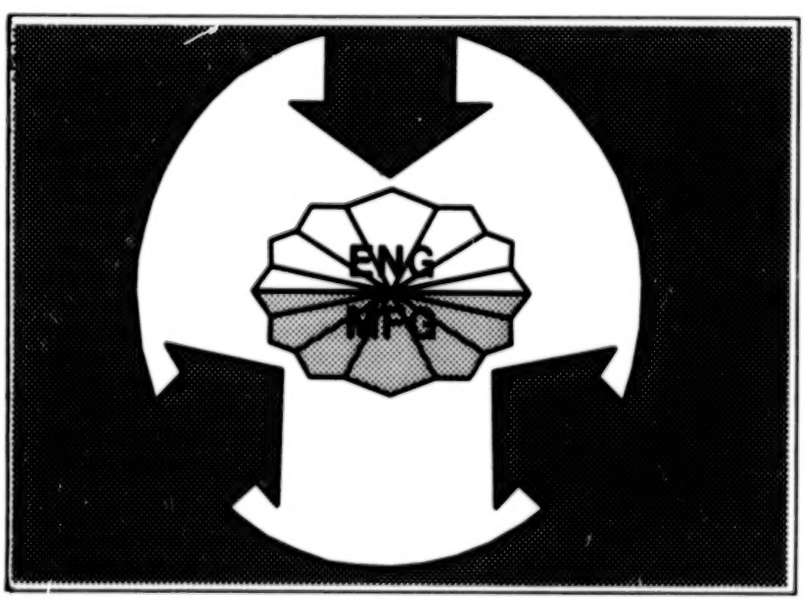
Engineers operating in this environment appreciate this integrated environment because it frees them to be creative without

having to learn the mechanics of the system. These details are automatically performed by the software and the data administrator. These executives, however, are typically file-oriented rather than function-oriented and can become administratively troublesome if not carefully controlled.

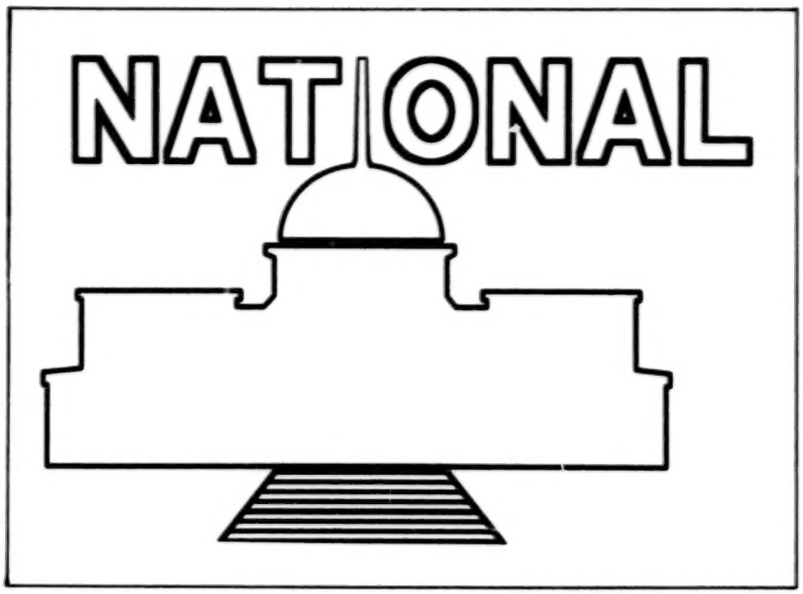
Recognizing the problems associated with the above method of integrating application programs, some companies are looking at data base management as the key to integration. The authors concur in the data base management approach but caution that DBMSs can also lead to the creation of isolated islands of computing power. Communication between islands is essential for overall company integration.



Along with the trend in other industries, aerospace is now in a heterogeneous distributed engineering/manufacturing data base management environment. Both the hardware and the software are heterogeneous, but there are still few, if any, data management systems that meet these requirements.



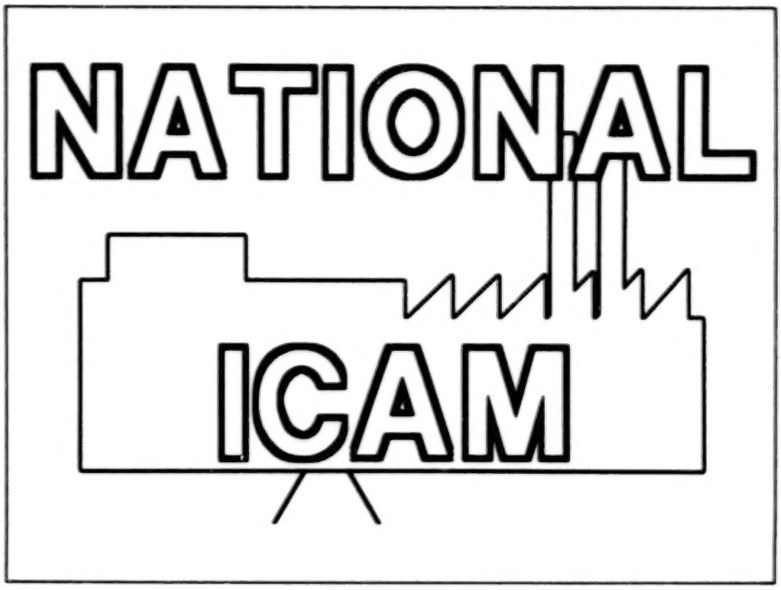
Additional areas of research need to be explored. One of these is distributed data base management in a heterogeneous machine and data manager environment. The IPAD program is engaged in this research and will be prototyping a very small segment of this system during fiscal year 1984. Today, companies around the United States should be studying their own data management problems for the benefit of their engineering/manufacturing complexes. Pilot environments should be investigated and the concept of backend data base machines considered. Other areas using DBMS technology, such as local versus global data, should be considered. Questions arising from these areas must be answered to meet the need posed by CAE technology.



Several other Government-funded projects are looking at integrating information flow, most of them focusing on studying needs, defining standards, and demonstrating prototypes. The variety of hardware and software systems being used by industry makes it virtually impossible to create a totally general-purpose information flow system that will satisfy every need. Each company will need to do some integration work on its own, directed toward its specialized programs, even if vendors provide hardware and software interfaces. Thus, the emphasis seems to be on defining standards and interfaces.

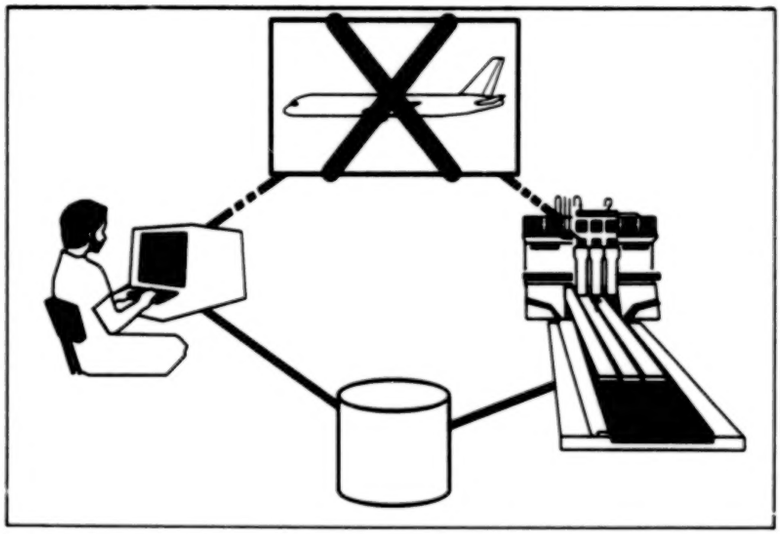
One such standard for transmitting geometric data between systems is initial graphics exchange specifications (IGES), developed by the National Bureau of Standards and now under continual evaluation for update as new technology (e.g., solid modeling) requires. Each vendor must provide translators to transform data from its own format to the standard IGES format and vice versa so that IGES files may be transmitted between different systems and even different companies. The IGES standard is being accepted by many companies, and translators have been or are being written for various systems.

Another project at the National Bureau of Standards is the Automated Manufacturing Research Facility (AMRF), which completed the first step in the integration of its small batch manufacturing facility in November 1983 (ref. 8). AMRF is creating a research facility to be made available to industry and universities working on the automation of small batch manufacturing. They are using DBMSs and communication protocols to interface between operators, robots, and machines with the goal of establishing interface standards that vendors will implement. This will enable companies to build manufacturing facilities with heterogeneous hardware and have machines communicate with each other more easily than is currently possible.

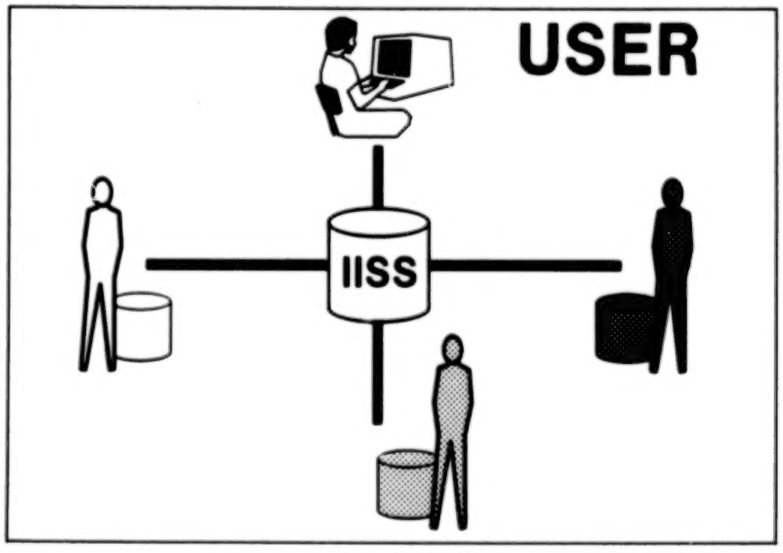


The Integrated Computer-Aided Manufacturing (ICAM) program also has a number of studies under way, looking at the integration of manufacturing facilities. The ICAM Integrated Sheet Metal Center is defining the requirements of a computer-based information system (CBIS) (ref. 9) and presently is defining conditions to be met by a hypothetical CBIS. It is expected that the resulting technology transfer with the greatest impact will be the standards described and employed by CBIS, which will be concerned with all aspects of information exchange within manufacturing.

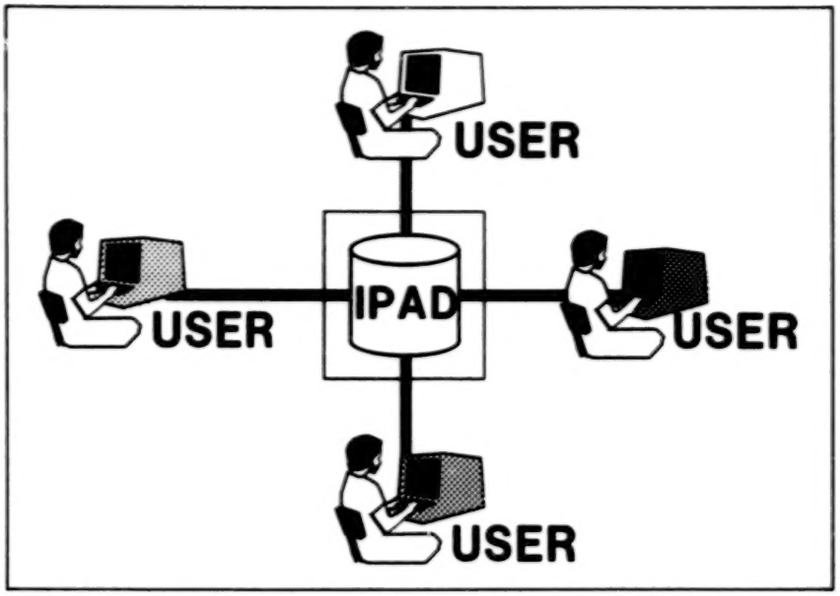
The ICAM Factory of the Future project (ref. 10) is also looking at communication within the factory. This project has done a state-of-the-art survey in various disciplines involved in information/communication in the factories, including DBMSs, networks, and hardware and software interfaces. They have documented many of the voids that exist in today's technology.



The Product Definition Data Interface program (PDDI) (ref. 11) being developed at McAir is to take the place of blueprints, serving as an information interface standard between engineering and all manufacturing functions using a blueprint (process planning, NC, QA, tool design, etc.). PDDI plans to define the interface, develop a software system, and perform a demonstration by the end of 1985. They will be transferring data between CAD systems and a number of manufacturing applications.



The Integrated Information Support System (IISS) (ref. 12) project is working with heterogeneous hardware and data managers, developing a system that insulates the user from having to know which system the user's data resides on. The long-term goal is to provide an environment that makes all the computers look like one integrated computer, with all the data appearing to reside in one data base accessed by a single, consistent type of terminal interface.

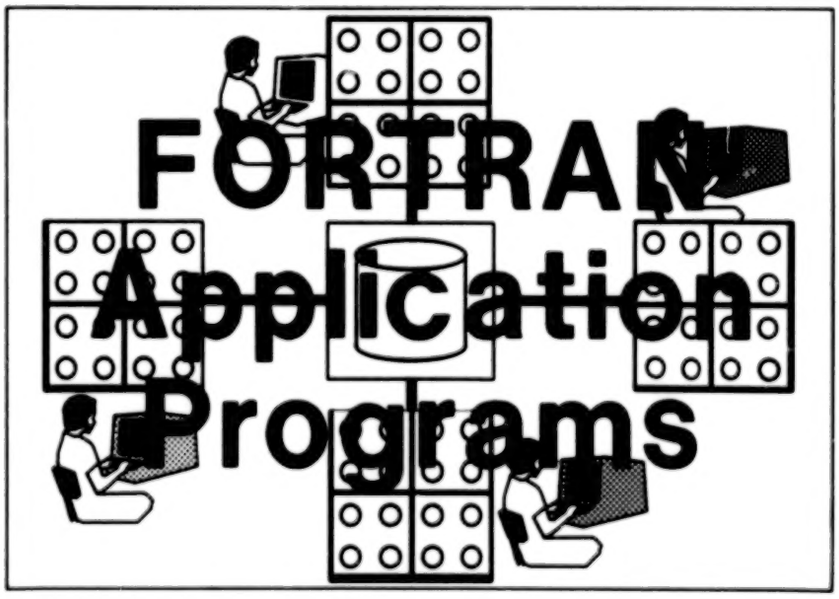


A PARTIAL SOLUTION TO MEET THE NEED

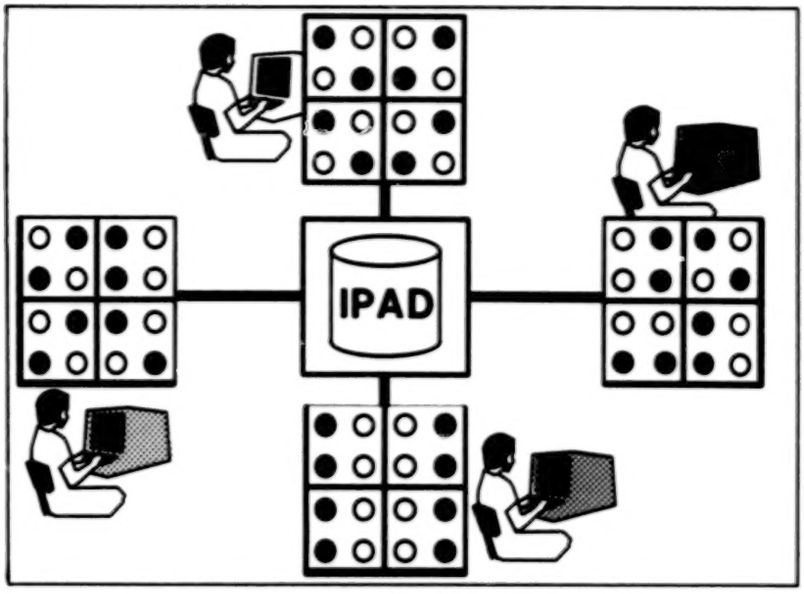
IPAD has developed a prototype data base management system that addresses the information management problem in the engineering/manufacturing environment. Version 5.0 of the IPAD information processor (IPIP), released in December 1983 (refs. 13,14), is a multi-user data base management system operating on the CYBER. Network facilities developed allow an application residing on the VAX to communicate with IPIP and the data base on the CYBER.

IPIP supports multimodels: relational, hierarchical, and network. It employs a multischema architecture to allow different users to have their own view of the same physical data. There is one family of schema languages (for internal, logical, and mapping schemas) for all the data models. The physical data description is described in an internal schema. Logical views of the data are contained in logical schemas. Mapping schemas define the relationship between internal and logical schemas, or two logical schemas, thus allowing for n levels of schemas.

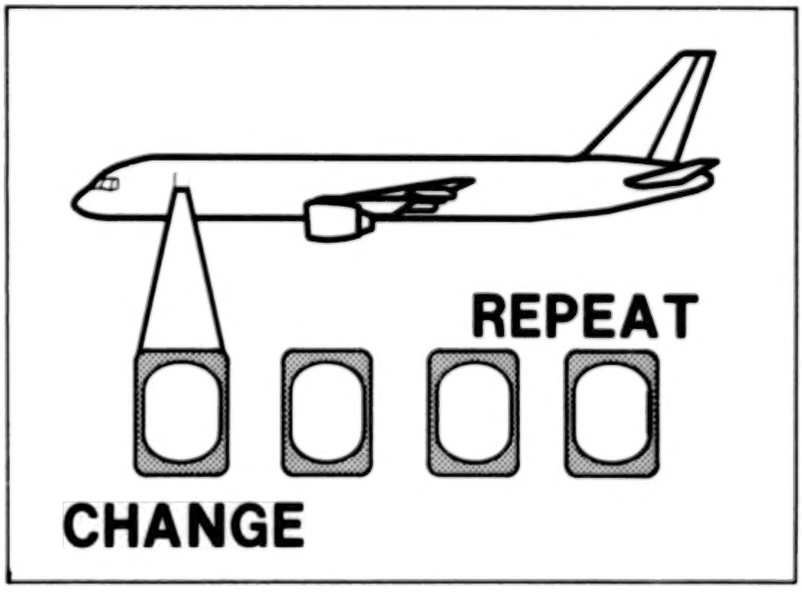
IPIP allows for the definition of scientific data types, including vectors and matrices. When data is defined in the relational model, it looks very much like tables of data currently used by engineers.



Users have access to the data through FORTRAN application programs, which can be written against any logical schema. One data manipulation language (DML) is used for all data models. DML statements are inserted into the application program, which is then precompiled, transforming the DML into valid FORTRAN statements. DML statements allow the user to query and update the data base.



A mechanism exists for restricting a user's window of data in the data base. A data set is associated with each row of data stored in the data base. A DML command exists to unlock data sets for read or update. Thus, a user has access only to the data in data sets available to him. Data sets may span relations and data bases (schemas). IPIP also allows for the definition of another pseudo data model—complex entity relationships (structures). This combines the relational and network models to manage complex data relationships. Some extensions were made to the DDL and DML to allow for this. This concept has been applied to the management of geometric data.



IPAD has defined schemas using structures that help to maintain the integrity and reduce redundancy of geometric data. We chose a particular geometric form, but others could use the same types of constructs to model their own geometric form. No change to the data manager is necessary to do this.

The entity types currently supported by the IPAD geometry schemas are points, lines, circular arcs, and objects. Objects are mechanisms for grouping entities, including other objects. Owner/member relationships are defined on the relations in the schema to help maintain the integrity of the data. A line cannot exist, for example, if its constituent vertices do not exist. Constraints on the relations prevent such things as a point being deleted if it is being used on a line. The structure of the schemas allows for the sharing of entities. A point can be used on several lines; for example, one command issued to change that point would result in it being changed for all the lines it is used on.

The same DML is used with processing complex entity relationships as is used with regular record processing, though not all commands have been implemented in Version 5.0 for the former.

IPIP VERSION 5.0 UTILITIES

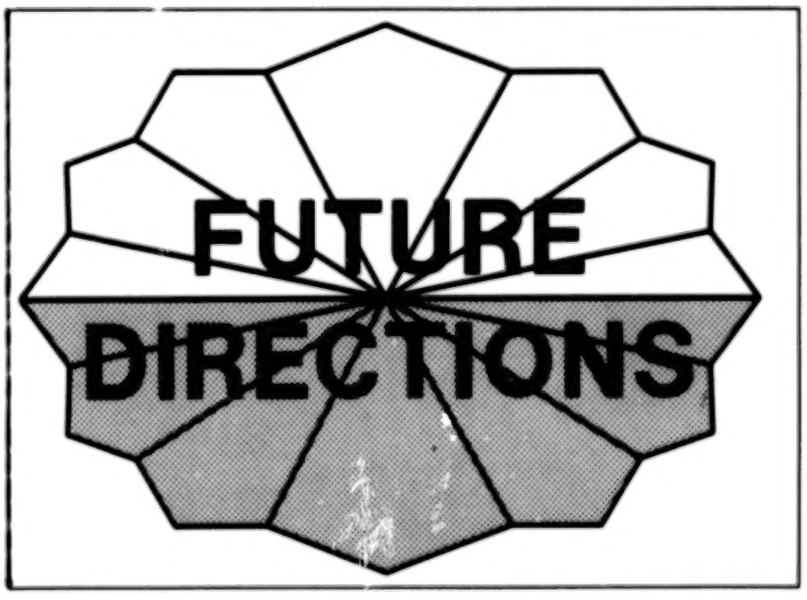
- IGES TRANSLATORS
- INTERACTIVE QUERY FACILITY
- SINGLE USER VERSION

Version 5.0 of IPIP also includes several utilities. Translators have been written to allow the interface between the IPAD geometric format and the IGES format. The translators handle all the entities implemented in IPIP. Thus, geometric data stored in IPIP can be transferred to some other system that can read an IGES file and vice versa.

An interactive query facility was developed to allow the interactive query of data stored in IPIP. This facility extracts a requested data set/record interaction from IPIP and transfers it to a corresponding schema in RIM (Relational Information Manager). The RIM query may then be used to selectively retrieve the data.

A single-user version of IPIP has been developed to allow users to experiment with the functionality of IPIP without having to install the network and IPIP at individual control points. IPIP/SU (IPAD information processor/single user) has all the functionality that IPIP has, including complex objects, except for the bulk record-processing facility. It operates only in a single-user environment.

The IPAD project has produced prototype software to provide a company with a research vehicle designed to solve a specific problem.



AREAS OF FURTHER PROTOTYPE WORK TO MEET COMPANY NEEDS

We have enumerated the various capabilities of the IPAD information processor. Given its research and development orientation and resource limitations, IPIP provides only a partial solution to meet a company's CAE data management requirements. Some of the areas where further work on IPIP would be required to more fully meet the CAE requirements include:

- Distributed data base management capabilities
- Extensions of data set facilities to provide configuration management capabilities (e.g., facilities for versioning, releasing, and archiving data sets)
- Additional geometry capabilities
- Recovery facilities
- An interactive query interface
- More friendly user interface

The IPAD project will be addressing some of these areas this year. Beyond that, there are many opportunities for industry involvement in IPAD yet to be explored.

REFERENCES

1. "Reference Design Process," IPAD document D6-IPAD-70010-P, The Boeing Company, Seattle, 1977.
2. "Product Manufacture Interactions with the Design Process," IPAD document D6-IPAD-70011-D, The Boeing Company, Seattle, 1977.
3. "Manufacturing Data Management Requirements," IPAD document D6-IPAD-70038-D, The Boeing Company, Seattle, 1980.
4. "IPAD Requirements," IPAD document D6-IPAD-70040-D, The Boeing Company, Seattle, 1977.
5. "IPAD Preliminary Design," IPAD document D6-IPAD-70036, volumes 2 through 9, The Boeing Company, Seattle, 1980.
6. "First-Level IPAD User Requirements," IPAD document D6-IPAD-70016-D, volumes 1 through 3, The Boeing Company, Seattle, 1980.
7. "IPAD: Integrated Programs for Aerospace-Vehicle Design," proceedings of IPAD National Symposium, NASA Conference Publication 2143, September 1980.
8. Charles McLean, Mary Mitchell, Edward Barkmeyer, "A Computer Architecture for Small-Batch Manufacturing," IEEE Spectrum, May 1983, pp. 59-64.
9. "Information System Requirements for ICAM Sheet Metal Center," ICAM Project Priority Final Technical Report, AFWAL-TR-82-4058, volume 1, AFWAL/MLTC, Wright-Patterson AFB, Ohio, July 1982.
10. "ICAM Conceptual Design for Computer-Integrated Manufacturing—Task B, Establishment of the Factory of the Future Framework—State-Of-The-Art Document," ICAM Project Priority 1105, document SAD110512000, AFWAL/MLTC, Wright-Patterson AFB, Ohio, September 1982.
11. "Project Definition Data Interface," interim technical report, ICAM Project Priority 5601, AFWAL/MLTC, Wright-Patterson AFB, Ohio, January 1983.
12. "Integrated Information Support System (IISS)," proceedings of the Seventh Annual ICAM Industry Days, ICAM Project Priority 1701, U.S. Air Force, Wright Aeronautical Laboratory, June 1983, pp. 371-394.
13. "Data Base Administration User Guide, IPAD Information Processor (IPIP) (Version 5.0), CYBER/VAX Configuration," IPAD document UM-REL5-200, The Boeing Company, Seattle, December 1983.
14. "Application Programming User Guide: Interfacing and Integrating Application Programs (Version 5.0, CYBER/VAX Configuration)," IPAD document UM-REL5-300, The Boeing Company, Seattle, December 1983.

CAD/CAM AND SCIENTIFIC DATA MANAGEMENT AT DASSAULT

Pierre Bohn
Avions Marcel Dassault-Breguet Aviation
Paris, France

ABSTRACT

The first part presents the history of CAD/CAM and scientific data management at Dassault. Emphasis is put on the targets of the now commercially available software CATIA. The links with scientific computations such as aerodynamics and structural analysis are presented.

Comments are made on the principles followed within the company. The consequences of the approximative nature of scientific data are examined. Consequence of the new "history" function is mainly its protection against copy or alteration.

Future plans at Dassault for scientific data appear to be in opposite directions compared to some general tendencies.

INTRODUCTION

As an introduction, I will give some figures about my company, the complete name of which is Avions Marcel Dassault-Breguet Aviation since a merger in 1971. The size of the company is about 16 000 employees. The design office is about 1500.

The important figure related to this conference is the number of families of aircraft active in 1984:

- 5 military aircraft families

- 3 business aircraft families

These 8 families are in production and 4 other programs are under development. Within each family, many different versions can exist, due to a high export activity.

These 12 active different programs represent an important amount of information to manage, and it is obvious that this can be a tough problem, especially when you relate this number of active programs to the size of the company.

Just a word about nonscientific data management. I include under this work activities such as purchasing, spares management, and so on, starting with the part list established by the design office. This data management started 10 years ago under IBM/IMS and is developed as a company-wide system by a unique team. Several hundred alphanumerical terminals installed in different places in France are linked to a computer center located near Paris and dedicated to these IMS data bases. Being not involved in this activity, I will now focus on design and manufacturing data.

SHORT HISTORY

When we started using computers - which for us was in the late 1950's with the Burroughs E 500 and the IBM 650 - it was mainly for performance computation, and organized data were in tables. Then, with the generation of the IBM 360, data were organized in files. Applications were mainly structural analysis and flutter prediction.

In the middle 1960's we started theoretical aerodynamics computation. This use of computers was - and still is - a necessity for our company. The amount of money that we receive for a development program of a new military aircraft is much less than what appears to be in development contracts in the U.S. As an illustration, I have seen many times that we could only spend, for example, 5000 hours of wind tunnel time when equivalent programs in the U.S. had used about 15 000 hours.

To be able to produce competitive aircraft, we needed to be able, with theoretical computations:

- to eliminate solutions which do not present a high probability of success

- to help to understand problems encountered in wind tunnels and converge more rapidly towards the best solution

So we created a strong team of aerodynamicists and mathematicians, and the result is that in some cases, such as transonic computations of the aerodynamic flow around very complex shapes, we think that we are in advance of the worldwide scientific community (ref. 1).

A necessary tool for these aerodynamic computations is the possibility of introducing in the computer the definition of complex surfaces. The first code for geometric definition arrived in the late 1960's as a batch program and was named DEDALE.

A pure 2-D tool, specialized at the beginning for wind tunnel models and running interactively on a graphic display, appeared in 1971. Its name is CALIBRE and it is still in use in some particular cases.

The second generation of the 3-D geometry (still in batch) was written in PL1 instead of Fortran for DEDALE and was used from 1973 up to 1979, when the interactive 3-D code arrived (CATIA).

In 1974, we were worried by the amount of work necessary to go from our geometric data files to the files necessary for our numerical control machine tools, programmed with APT language. As we were - and still are - the greatest user in Europe of large 5 axis NC machine tools, this was a problem of efficiency. We wanted to link more closely design and manufacturing by unique software tools. We proposed to our general management the choice between two solutions:

(a) Buy an existing code and modify it to match our most important needs, especially in the 5 axis NC field.

(b) Develop internally a software tool meeting completely our specifications, with an estimated time of 3 years.

The decision was to do both. We installed the Lockheed product CADAM starting in 1975 with 4 graphic displays. We wrote, from 1975 to 1977, 25 000 lines of code matching CADAM to our needs, and we later sold them back to Lockheed.

The 3-D interactive tool, named CATI and then CATIA, was first used in 1979 to design and manufacture high speed wind tunnel models.

In 1978/1979, we bought several Computer Vision systems mainly for electrical wiring and avionics schematics.

In 1980, important decisions were made. As we are working with many subcontractors and as in Europe many aircraft programs, civil or military, are shared by several countries, we thought that, whatever the qualities of our internal product, we would be obliged to use in some years one of the worldwide accepted standards of CAD/CAM. So, in order to avoid losing investment costs and to pay once more for the training of several hundreds of people and to transform eventually our old data into new formats, we decided to be a candidate for one of these future industry standards. Discussion started with several companies. One of them, and not the least, put us in a technical competition with other products coming from U.S. and Japan. We are very proud to have been chosen by IBM to complete their CAD/CAM catalog upwards in the 3-D domain. The nonexclusive license agreement between Dassault and IBM was announced officially in November 1981 in Detroit.

In December 1981, the team which developed CATIA left the Dassault Company to form a subsidiary named Dassault Systemes. Their responsibility is to develop and support worldwide the CATIA product.

The links between the mother company and the subsidiary remain very strong. The needs of the mother company in the CAD/CAM field have to be satisfied by Dassault Systemes and the new modules are tested within design office and plants of Dassault before outside installation.

Among these new modules, one which will be put on the market this year consists of an automatic mesh function for structural analysis. We have developed for internal use several generations of code for structural analysis. The last one, based on finite elements, is about 10 years old. Its name is ELFINI. ELFINI is advanced compared to existing available codes for structural analysis mainly in the fields of optimization, carbon fiber structures, and crack propagation.

ELFINI was working before CATIA. An interface with CATIA has been realized in 1983 and it is that interface which will be put on the market very soon. It will provide CATIA customers a very fast and powerful interactive way to build a mesh usable later by codes like NASTRAN and others.

This way to proceed will be discussed later in this paper.

A robotic module has been recently announced.

PRESENT SITUATION

The total number of graphic displays for the Dassault Company is this year around 200. The past years' average increase of this number was around 40 percent per year.

Links exist between CATIA, theoretical aerodynamic computation and structural analysis. CATIA is around 400 000 lines of code, and our subsidiary dedicated to CAD/CAM software will be shortly above 100 employees. An important part of the aerospace and automobile industries is now using CATIA in the U.S., in Japan and in Europe.

COMMENTS ON HISTORY

Besides the figures and facts presented above, some comments can be added.

Within Dassault, scientific software development is very decentralized, contrary to the nonscientific software. Codes are written by engineers themselves and these engineers are not in a unique department.

With one exception, all this scientific software is developed inside the company and is considered as a part of our know-how. This exception concerns CAD/CAM. In this domain, we imported and are now exporting lines of code. My explanation is that in the CAD/CAM activity there are many aspects similar to a language, and that is stronger than the internal know-how aspect.

Another comment concerns the 2-D and 3-D way of evolution. As I said, the need in the mid 1960's was to have digitized information about the geometric definition of our aircraft to be able to compute aerodynamic flows. So we started directly at the 3-D level and with the most complex definition of surfaces. This approach was different from several CAD/CAM systems, which started with the capability of designing simple objects and has been upgraded slowly, with discontinuities and sometimes with incompatibilities. In our case, our interactive system was operational with complex surfaces in 1979. The 3-D approximation of polyhedral surfaces, necessary to make in a sufficiently short time hidden line removal or Boolean operations between volumes, appeared in 1981. Finally, basic 2-D functions will be available this year. This kind of top-down approach, even if it was motivated by internal priorities, was very useful for starting from the beginning with a coherent system and avoiding, during the development, destroying or changing basic concepts.

SCIENTIFIC DATA ORGANIZATION

I have already mentioned links between external shape data in CATIA and aerodynamic computations and between geometrical definition of mechanical parts designed with CATIA and the structure analysis. There are many other internal links within the company.

For the main design office located near Paris, the scientific data stored on line represented at the end of 1983 about 30×10^9 bytes and it is continuously increasing. I would not dare to say that this large amount of information is a clean, well organized data base. Of course, this depends on the definition used for a data base. Personally, I think that for our nonscientific data, which is almost completely well organized with IMS hierarchical structure, the word "data base" can be used. But the case for scientific data in Dassault can be described in general as a set of private files linked by interface files. This will be described by examples.

The theoretical aerodynamic team, when they optimize external shapes in subsonic and/or supersonic, have as a side result of their computations the complete pressure

distribution around the aircraft body. The structural analysis team can use this information after appropriate integration to have values of aerodynamic loads used as inputs for their structure computations. With these inputs, they can compute with their structural model, for example, the modified shape of the wing, send it back to the aerodynamicists, who can compute new pressure distributions, and so on. This loop is relatively slow, both types of computation (aerodynamic and structural analysis) being themselves very long with the present level of power of computers.

Sometimes, for example at the preliminary design level, we can accelerate the process by simplifying one of the two components of the loop. A simplified model of the aeroelastic wing can be used by aerodynamicists or a simple linear model of the aerodynamic flow can be an input for the structure team.

The intellectual process which allows a team to provide to another one a simplified model is very general. Many other examples can be given. Experimental data, such as an unstationary pressure measurement in front of a jet-engine compressor, can be transmitted after a given filtering as an average value and a rms component. One can see that these processes have in common a sender and a receiver and that a dialog is necessary between them. The receiver must explain his needs; the sender must, before transmitting the information, provide the domain of validity of this simplified - or reduced - information.

I have avoided up to now to use the word "approximation" to emphasize now this key-word. The scientific world - our "universe of discourse" - is an approximate world. I will come back to one example given above. Starting from an accurate polynomial definition of surfaces, the aerodynamicist will approximate the surface by a grid; he will build a mesh for the volume around the body using one of the possible discretization techniques, the minimum size of an elementary volume being imposed by the power of the computer and the time available. Then he will use an approximation of the Navier-Stokes equations, with a simplified model of turbulence. Many times, he will simplify a lot by using completely inviscid flow equations. The results obtained after a certain number of iterations converging towards an approximate solution of the problem will be then transmitted to the structural analysis specialist who will compute with these aerodynamic loads a structure approximated by finite elements, and so on.

The very great number of levels of approximations makes me think that our scientific data are of a different nature than business data. My number of children appearing in the employees data base of my company or the number of a certain type of bolt stored in one of our plants are not approximated values. So this information can be assessed by people having of course the correct access rights without a direct relation between the producer and the consumer. This is why it is not evident to me that solutions of shared data bases are always adequate for our kind of scientific data.

GENERAL SEMANTICS

I cannot resist to the temptation to speak about the parallel which can be made with the general semantics theory. I have read about this theory in reference 2. I am sorry to make reference to a book written in French but I give at the end of this reference a reference to a U.S. book which I had not the opportunity to read. The father of the general semantics theory is Alfred Korzybski, who was born in Poland and lived in the United States from 1921 to 1950. There are many very interesting ideas in this theory.

One of them insists on the importance of the abstraction process - "An abstract concept is generally an idea which neglects details." Events, objects, names of objects, names of families of objects and so on are different levels of abstraction. "Human knowledge has been able to increase only by voluntarily forgetting details related to individual cases."

The approximations that I mentioned before are also steps where details are forgotten.

Another idea in the general semantic theory is to avoid the structure of the language which, since Aristotle, considers the world as static while reality is dynamic. There is between human beings a continuous exchange of messages which modify the apparent world.

About this transmission of messages, it is said that one of the basic principles is that a message must be elaborated in function of the receiver. In other words, in our world of computer science, it is often said in the several models of distributed or parallel computations that communications between processes can be made by shared data or by messages. So, going back to the organization of the scientific data of my company that I shortly described above as a set of private files linked by interface files, I would like to consider these interface files as messages. Some of these interface files represent the same information at different levels of approximation because they are used by different receivers for different purposes. An internal standard for these interface files has been proposed to the users and is more and more accepted to help communications between teams.

When communicating, teams have to use a bilaterally agreed upon language, but it is not necessary:

- that this language has to be the same for every exchange (concept of type of messages)

- that the same data representation has to be used by every team (concept of private objects)

To impose a unique model for very different scientific activities can be a tremendous and expensive task, with a risk of losing the very useful adaptability and flexibility of the human beings. On the other hand, it can be important to try to be sure that messages are correctly understood, that dialogs are well established, that the risks inherent to the chosen level of approximations are agreed to by the producer and the consumer. This can be better achieved within a decentralized structure limited to what is strictly necessary.

CONFIDENTIALITY

An interesting feature of CATIA is its "history" function. This function is used within the company in some limited applications. It has not reached yet a sufficient level of generality to be put on the commercial market. But the consequences of this function can be interesting to discuss.

The principles of the history function are simple. We store not only the result of a design in the form of a model but also all the actions made with the function keys, the light pen or the keyboard by the designer during the design phase. We store the result of the question what plus also the answer to the question how.

The main purpose of this function is to reduce costs of modifications by permitting, for example, playback of the design up to the point where the modification is introduced and eventually continuing automatically the design. This function can be used to keep the expertise of an engineer before he retires. And, as the know-how of a company can be considered as the sum of the individual know-how's, it is important to be able to restrict the access to the information representing that know-how. It can be more important to protect this know-how than the definition of one specific product.

This importance can exist outside of defense-related activities. I think that "history" storage is a normal evolution of computer-aided engineering and then that we play more and more with fire. Limited access rights, passwords, and encrypted data are efficient means of protecting information, but only up to a certain level. It is a problem of relative cost between the value of the information and the costs and risks of breaking the protection lines. We will increase the value of our scientific digitized information, so we will have to increase the protection.

Stored and transmitted data have to be protected not only against copy but also against alteration. I am speaking of course about voluntary alteration. Vulnerability of companies is increased by the proliferation of transmission lines which become every day more and more necessary for the life of the company. Duplication of lines is a good answer to accidental alterations but it is not sufficient for the cases we are worried about.

Large shared data bases imply transmission lines towards alphanumerical or graphic displays. This can be an efficient and economical solution for commercial data management but it is clear for me that this kind of solution can be, for my aerospace company, dangerous if it is used everywhere for every kind of data.

CONCLUSIONS

I tried to present earlier in this paper some reasons for the model of organization of our scientific data (private files plus interface files). It also seems to fit with the problem mentioned just above, and I now see for my company the following evolution of our scientific tools.

Private data will tend to be stored either in highly protected locations or on movable supports which can be stored in a safe near a work station. These movable supports can be small disc cartridges or, in a very short time, optical disks. These cartridges will be preferred to transmission lines to transmit sensitive information. They will be the normal support of interface files when sender and receiver are not in the same location and they will have during transmission a protection equivalent to classified reports. Why not keep the basic rules admitted for these classified reports? They have a limited and known list of receivers. At the reception of a protected physical support the receiver gives an identifiable acknowledgment.

You can see that such a tendency is in an opposite direction compared to the general evolution towards large shared data bases transmitted through several levels of networks to many terminals. To be sure to avoid any misunderstanding, I do not conclude that this general evolution is wrong. It is certainly very efficient when applied on a university campus to distribute knowledge and it is evidence that the type of wired (or cabled) organization will be more and more used. It is no reason to apply this general model to our scientific data if we have several other reasons to make different choices.

REFERENCES

1. Heckmann, G.: Etude par la méthode des éléments finis des interactions voilure-fuselage-nacelle d'un avion de type FALCON à Mach = 0.79. AGARD Conference Proceedings no. 285, 1980.

2. Saucet, M.: La Sémantique Générale aujourd'hui. Within this book, there is a reference to A. Korzybski: Science and Sanity, 1933.

D9

DISTRIBUTED DATA BASE SYSTEMS WITH SPECIAL EMPHASIS TOWARD POREL

Erich J. Neuhold
Hewlett Packard
Information Management Laboratory
Palo Alto, California

ABSTRACT

In the last few years a number of research and advanced development projects have resulted in distributed data base management prototypes. POREL, developed at the University of Stuttgart, is a multiuser, distributed, relational system developed for wide and local area networks of minicomputers and advanced micros.

In this paper the general objectives of such data base systems and the architecture of POREL are discussed. In addition a comparison of some of the existing distributed DBMS is included to provide the reader with information about the current state of the art.

1. INTRODUCTION

During the last few years distributed processing has gained much attention in large governmental organizations and large private enterprises faced with the problem of running and controlling an ever increasing number of geographically distributed computing facilities. The increased use of workstations and intelligent terminals also calls for a coordination of what is to be stored in a system and how it is to be used.

Many different fields of data processing have to be investigated when trying to integrate the various systems in a network such that a user will be offered a single common view of the software, information storage, and processing facilities available. For example, it should not be required that a user of such a system has to know about the idiosyncrasies of some specific operating system running on one or the other of the machines in the network. Currently such specialized knowledge cannot be avoided and has added considerably to the problems associated with distributed computing facilities.

Among the areas that have to contribute towards a solution are:

- o communication systems and communication protocols - for example, those contained in the open systems interconnection proposal of ISO [1]
- o data representation in and data conversion between the different machine architectures involved,
- o common command and control languages interfacing with the different operating and filing systems,
- o distributed processing with load balancing and special purpose machine utilization features,
- o distributed data storage with controlled redundancy, consistency and recoverability,
- o intelligent unified user interfaces to guide the different users, e.g. system administrators, application administrators, managerial and technical end users, through the many facets of such distributed application systems.

Practically all of these aspects are included one way or another in a distributed data base management system. A number of such systems are now in the stage of advanced prototypes and some of them are actually available on the market. Among those systems are

SDD-1, MULTIBASE, ADAPLEX
(Computer Corporation of America, Boston) [2,3,4]
R* (IBM Research, San Jose) [5]
DISTRIBUTED-INGRES (UC, Berkeley) [6]
SIRIUS (INRIA, Paris) [7]
HERMES (Italy) [8]
POREL (University of Stuttgart) [9]

In the following we shall describe the general objectives and the architecture of POREL and later include a comparison with some of the above mentioned systems.

2. THE OBJECTIVES OF POREL AND ITS ARCHITECTURE

2.1 Objectives and Strategic Decisions

When developing distributed data base management systems, two basically different approaches exist.

1. Build a system which integrates centralized data base systems already in existence at the various nodes of the network to produce a single distributed data base system. In this case, the principal problem is to integrate sometimes highly different data models which may have been constructed using different data modeling principles, e.g. hierarchical [10], network [11], relational [12], and different design strategies. For a number of years repeated efforts have been made to build such systems, but so far no general system is available. The possible differences in data structuring are such that it will probably always be impossible to integrate the individual systems automatically. Human expert help will be needed to provide information which unfortunately sometimes exists very implicitly only in the data structures, or depending on the model may be present in the various application programs. The closest one to a solution using this approach is Multibase [3] of the Computer Corporation of America.

2. Build a new system from scratch which provides a unified approach to data representation and manipulation. Different data models could still be used but they would have to be translatable into each other and therefore the full set of features of any individual data model would probably not be utilized. As a consequence, practically every such system only provides a single approach to data modelling and in addition assumes that data already existing in some centralized data base will be reloaded and if necessary even converted for the distributed data base system.

With the exception of Multibase all the distributed data base management systems mentioned in this paper have chosen the second alternative. For POREL we have chosen this alternative since we wanted to present the users with a relational data base model and considered the problem of conversion between data models to be a completely separate and very difficult research subject.

After selecting the principal approach we had to make many additional decisions which later were formulated as objectives for the POREL project.

1. POREL works on a network containing computers of many different types and sizes, down to advanced microcomputers, and using many different operating systems.

   o This objective requires transportability and tightly controlled (simple) interfaces to the environment. In addition, it requires the software to be written in a higher level language where compilers are available on all the different operating systems. After some investigations we have chosen PASCAL as the language most widely available. PASCAL is not really a system implementation language and as a consequence some inefficiencies in data storage and manipulation result. In summary, however, we have been quite satisfied with our choice. To transport system components between different machine types has proven to be in every case quite simple. Because of the tightly controlled interfaces, adjustments to the various operating systems also have been

rather easy as long as some of our basic assumptions (see section 2.2) on the system are satisfied.

2. The network is widely distributed in the sense that long distances, that is, public communication lines are involved.

   o This assumption requires that POREL accept low communication speeds and relatively high cost of communication. Public communication facilities are and will remain for the forseeable future rather expensive if compared to the price of the mini- and microcomputer systems appearing at the various sites of a DDBMS.

   As a consequence our attention has been on developing mechanisms relying on minimal numbers of intersite control messages and on minimizing the size of intersite data transport. After investigating various available or proposed communication strategies we have chosen for POREL the ISO/CCITT packet switching facility X.25 [13] and have developed the higher level protocols in accordance with the open system interconnection proposals presently being discussed in ISO and CCITT [1]. X.25 was chosen mainly because it was clear at that time (1978) that X.25 would become the packet switching standard of many countries and especially of the German Post Office where it was introduced in 1980 as DATEX-P [14]. Since at the time our implementation started no X.25 systems were commercially available, we implemented X.25 on LSI-11's using memory sharing to interconnect the LSI-11's with the data base node processors, especially our own PDP11's at the Institute.

3. The system will work on mini- and/or micro-computers offering as a minimum a 16 bit addressing space inside a process.

   o Many minis and micros still are restricted to such an addressing space even though the real storage attachable may be much larger. POREL is now able to work in 64 K storage, but we had to structure the system into many cooperating processes and in addition program overlay had to be utilized extensively. This requirement has actually led to many difficult implementation problems and has decreased the efficiency of the system considerably. One could even say that an effective sophisticated distributed data base management system should only be realized on systems offering more than 16 bit addressing space to a process. Since today even many micros offer larger such spaces, the restriction could possibly be dropped. POREL has been implemented such that an increase in the size of the process space can quite easily be utilized by repackaging the system as we are currently doing for VAX machines.

4. The system should continue working even if arbitrary partial breakdown of sites and communication lines should occur. The system may continue working with somewhat reduced capabilities, but has to be able to recover fully when the breakdown disappears.

- o As a consequence, POREL cannot utilize any centralized control facilities. A breakdown of such a site would stop the whole system. POREL now includes mechanisms for distributed control of transaction execution, data storage, data sharing, data redundancy, authorization, recovery, and reliability. Transactions not directly relying on any of the failing components will continue execution uneffected by the breakdown.

5. The distributed data base management system will allow dynamic readjustments in the network. That is, sites and communication lines may be included and deleted during times the system is running and executing transactions.

   - o This facility has been provided in POREL based on the recovery mechanism of the distributed system as such a mechanism has to be able to handle/reintegrate nodes temporarily non-functional and accomodate/reestablish temporarily interrupted communication lines. Incorporating/deleting nodes and lines just represents a special case of the general mechanism.

6. The data model of the distributed data base system should be relational to offer maximum flexibility for data model adjustments and data independence of the programs.

   - o We have chosen a distributed relational data model that allows division of relations into non-overlapping subrelations, provides for multiple copies of such subrelations and allows distribution of the subrelations onto different sites of the networks. Notice that the restriction to non-overlapping subrelations does not have any external consequences, because overlaps can be "simulated" by using multiple copies of individual subrelations. It decreases, however, the internal effort necessary to update/delete data in the data base considerably. Copies of subrelations are handled in POREL in such a way that a user will always be able to see a consistent picture of the data base. This does, however, not imply that all updates are communicated immediately at update time to all the copies of a subrelation. At update time only those actions will be executed that will enable POREL to perform all the necessary updates before the respective subrelations are eventually used again.

7. The system is to be utilized by three different categories of persons:

   a. programmers using some progamming language which includes a data base sublanguage

   b. data base administrators and data base users manipulating the data base via a non procedural stand alone data base language

c. application builders, administrators and end users working with their specialized interactive languages on applications utilizing distributed data bases

- o POREL now offers three different interfaces. First, an application development and usage system is provided for the application environment. This subsystem shall not be further discussed in the present paper. The interested reader is referred to the publications [15] for detailed description of the system. Second, a nonprocedural standalone data base language is provided. It is based on principles similar to SQL [16] and supports both the data base administrators and the data base users. The language, called RDBL (Relational Data Base Language), is built around the transaction principle, where a transaction is a unit of work which, when applied to a consistent data base, will produce a consistent data base again. Third, PASCAL is provided as a data base host language, with RDBL extended by a cursor mechanism as the data base sublanguage. This interface has been realized using precompiling principles. That is, the data base sublanguage statements are translated by a precompiler into PASCAL subroutine calls. This differentiates our approach from the integrated approach realized in the PASCAL-R langauage [17].

8. The same transactions may be executed frequently by the system. Many applications rely on a rather fixed number of predefined operations, e.g. airline reservations, banking, order handling etc.

   - o To provide for efficient processing of such transactions, compiling principles instead of interpretations have to be utilized. POREL allows the compilation and storage of transactions. In order to ensure well behaved execution at later times, a checking mechanism has been included which guarantees that the data base has not been changed in a way which would invalidate the original assumptions contained in the transaction or incorporated by the system at translation time.

9. The system is to offer for the programmer or end user a unified view. That is, these users do not have to be aware where data or pretranslated programs are stored in the network or where the transactions will ultimately be executed.

   - o POREL includes mechanisms which will split transactions into subcomponents executable on different sites in the network. The subdivisions are performed such that transportation cost for data will be minimized. However, currently no automatic distribution/redistribution of data takes place. The data base administration has to specify how to subdivide relations and where to store the various copies of such subrelations. A later extension could automatize this process or at least interactively support the DBA in this task.

10. The user may specify different levels of data base consistency to be enforced for his accesses to a relation. POREL will ensure that during the execution of one transaction the transaction may

    1. read and update relations freely without any interference from other concurrently executing transaction - as a consequence, a user will always work with a consistent picture of the data base

    2. read subrelations freely without interference from other concurrently executing transactions, where these subrelations may either reflect the latest version available in the data base or may reflect some earlier version - as a consequence users may work with an old but consistent data base picture where repeated reads of the same subrelation will always produce the same result

    3. read subrelations freely without ensuring noninterference from other concurrently executing transactions - as a consequence repeated reads of a subrelations may produce different results and not even inside a single subtransaction can consistency be expected

    Case 1 of course should be the normal mode of work. Case 2, however, is important where one deals with widely distributed and redundant data that in itself should be consistent but does not have to reflect the latest update status. Mail-order-house catalog information could be such an example. Since the final order will have to be checked anyway, it is not necessary that the user should be aware of the latest status of availability of a product. Inside of a single transaction he should however deal with a consistent picture of his data. Case 3 will probably be used only when dealing statistically with some data or when the user is sure that interference from others cannot hurt because of the peculiarities of his application.

    This flexibility of course is mostly provided to allow a reduction in the administrative overhead inside of POREL in case full data base consistency is not required. That is, it allows the user to tell POREL that it may execute some transactions more efficiently than otherwise would be possible.

2.2 The Architecture of POREL

In Section 2.1 we have developed the objectives and basic strategic decisions used in the development of POREL. We now want to investigate the architecture of the system. Here we can distinguish between the logical structure of the system and its actual implementation utilizing multiple cooperating processes.

In Figure 2.1 we have illustrated the logical structure of the system.

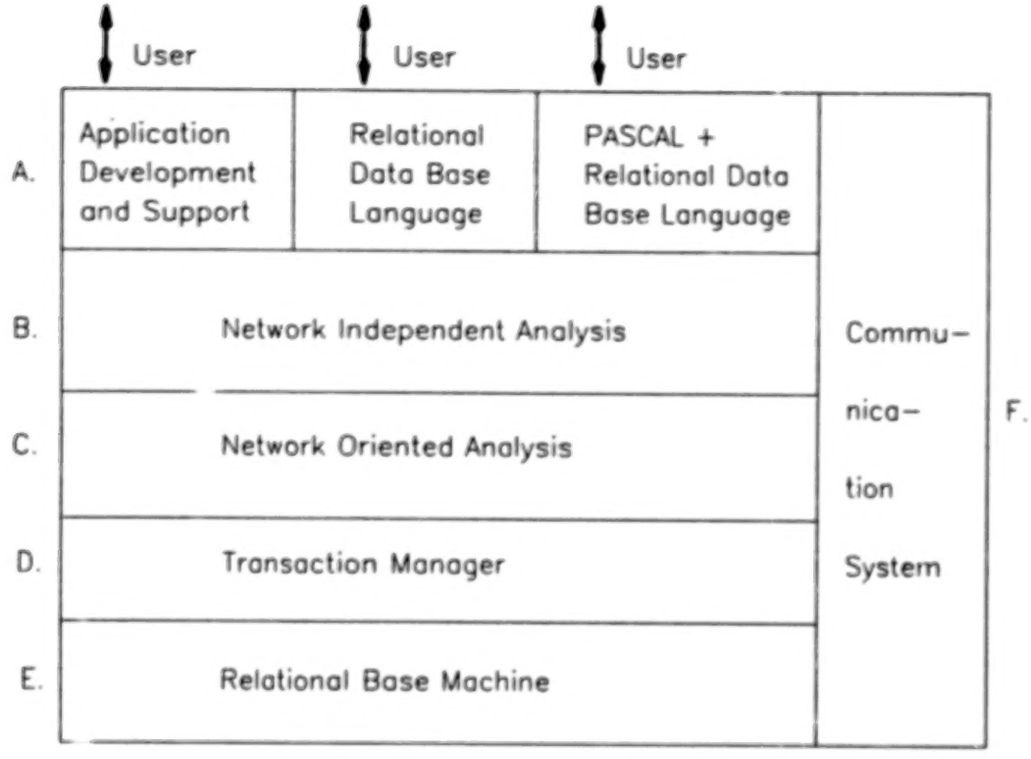


Figure 2.1: The Logical Structure of POREL

A. The different users have already been introduced in Section 2.1. For each interface a dialog oriented support facility is provided. In case of the Relational Data Base Language (RDBL) interface the user may specify interactively RDBL transactions. A transaction is the basic unit of work in POREL. It has the important property that whenever an execution is attempted, the transaction is either executed completely and the resulting data base state satisfies all given consistency constraints (an ASSERT statement is provided for that purpose) or it is not executed at all in the sense that it leaves the data base state unchanged. An RDBL transaction either consists of a single RDBL statement or it is represented by a sequence of statements enclosed in BEGIN TRANSACTION, END TRANSACTION brackets. In this paper we shall not discuss RDBL in detail and interested readers are referred to the literature [18].

The PASCAL • Relational Data Base Language (P-RDBL) interface offers to the user a procedural data base language. Using BEGIN TRANSACTION, END TRANSACTION the programmer may specify RDBL transactions inside of PASCAL programs. If no transactions are explicitely identified all RDBL statements contained in a PASCAL program are considered to form a single transaction. These transactions look essentially like the transactions specified when using the standalone RDBL interface with the exception that cursor definition and use statements may be incorporated and data exchange beteen PASCAL

variables and data base variables may be specified. Again the interested reader is referred to the literature [18], where a detailed description of P-RDBL is given.

In the Application Development and Support (ADS) system, the user can select, combine, recombine and execute different prepared programs from an application library [15]. These programs may be programs written in schematized P-RDBL which allows execution time substitution of relation names, attributes etc. In addition, a user may through his application language provide to these programs input data which are selected out of the data base or he may save program results in the data base. In all these instances the translations built into ADS will ensure that the data base interface is represented via RDBL or P-RDBL transactions. That is, for POREL ADS does not differ from any other application program or user utilizing P-RDBL or RDBL respectively.

As a consequence the lower layers of POREL only have to be able to handle RDBL (that is, P-RDBL) transactions. This has the additional advantage that ADS is transportable into environments where other relational data base systems are used. Only the transactions have to be produced differently; all the other components of ADS will remain unchanged as long as the interface languages into those data base systems follow the general philosophy of relational algebra or relational calculus languages with some navigational (cursor) facilities.

B. The Network Independent Analysis (NUA) takes the RDBL transactions (that is, the RDBL transaction parts of P-RDBL programs) and translates them into the relational algebra language RML which is considered the Relational Machine Language of the relational data base machines implemented via software in the POREL system (see point E below). The activities performed by the NUA have been selected such that NUA does not have to know about the network or the distribution of relations and subrelations in the network. In this respect it would work identically in a centralized data base environment.

C. The RML output of the NUA will be analyzed by the Network Oriented Analysis (NOA) subsystem of POREL. NOA utilizes the knowledge about network configuration, relation and subrelation distribution and redundancy to split transactions into substransactions where each substransaction can be executed in a single node of the network. The distribution of these subtransactions in the network is determined such that the data communication traffic between nodes will be minimized. NOA also produces the necessary RML statements for subtransaction distribution and data transport. The output of NOA, data base transactions in RML form, then may either be given to the user, stored in the system, or processed immediately.

D. The evaluation of transactions is initiated and supervised by the Transaction Manager (TM) which has to distinguish between P-RDBL transactions and RDBL transactions. In case of P-RDBL the TM module has to support the cursor features whereas cursors can never appear in RDBL transactions. The Transaction Monitor also is in charge of locking data, of scheduling the subtransactions for execution and of all the

recovery functions of POREL. It is important to note here that the functions of the TM are realized in a completely decentralized fashion. The TM at the site of origin of a transaction, i.e the site where the transaction was initiated, will be supervising the execution. Other TMs will also be involved in executing the transaction, but they only perform supportive actions for the supervising TM. This, however, may happen all over the network and the same TM may supervise some transactions and support others at the same time.

E. The Relational Base Machine (RBM) will be activated by the Transaction Manager for the execution of RML subtransactions. Since subtransactions by definition will be executed on a single node only, it is not necessary to build any knowledge about the network into the RBM. The RBM may receive data from or produce data for some other node, but this feature does not require any knowledge about the net as the Transaction Manager together with the Communication System will perform the actual transport. The RBM essentially works with these data the same way it would work with data given by the user of POREL. The Relational Base Machine processes the relational algebra programs expressed via RML, evaluates assertions and checks consistency, but also ensures that inserted updates do not violate the distribution criteria specified for the individual subrelations.

   In POREL traffic between nodes is always contolled by the Transaction Managers. This strategy is necessary since only the TM is able to react to failures in the network and can ensure the atomicity property of the transactions. All other POREL components only have a partial picture of the execution status of a transaction and therefore cannot decide on any recovery strategy when errors during transaction execution occur.

F. The Communication System (CS) is in charge of all data and message transport between the individual components - locally and network wide. The Communication System is based on the packet switching protocol X.25 [13]. Structured according to the communication system architecture proposed in the Open Systems Interconnection [1] it provides higher level commands which enable message and file transfer between the different POREL components even if different machine architectures and/or operating systems should appear in the network. The different layers of CS do not reflect the full OSI features. Only those components have been developed that are required in a distributed data base environment. The protocols are based on point-to-point communication with both communication processors taking active part in establishing the connections and executing the data transfer. That is, neither broadcasting facilities nor mailbox type delivery is directly supported by our communication system. More details can be found in [19] and [20].

After this discussion of the main components of POREL we present in Figure 2.2 the actual structure as reflected by the (principal) cooperating processes contained in POREL. But even here a word of caution is necessary. Because of address space limitations some of the processes shown had to be split into separate real processes but since these splits do not reflect major functionally separate POREL components, we have chosen not to identify them individually.

In our discussion of Figure 2.1 (the logical structure of POREL) most of the components have already been mentioned. There are, however, a few processes that require some further explanations.

- The processes marked in the left upper corner may only exist exactly once on a network node. For all the others multiple instances may appear allowing efficient multiuser support. Note, however, that the single processes Transaction Manager and Catalog Manager have been designed such that they can handle multiple users at the same time. Consequently POREL can be considered to truly represent a multiuser system.

- All the program and data flow between the processes is handled by the communication system CS. No direct communication between processes takes place.

- All communication between the sites in the network is performed via the Transaction Manager. This is necessary for the recovery mechanisms required to ensure the atomicity of the transactions in the data base.

- Data at one site may be communicated directly between local POREL processes, bypassing the Transaction Manager. This strategy was chosen for efficiency reasons, especially since even when using the TM the problem of atomicity of transactions as concerning the external world could not be fully ensured. This is especially true for the host language system. Partial results of a transaction will be communicated to the PASCAL program and used there, e.g. to write a salary check, before the transaction is completed. If the transaction now would have to be undone, this salary check or any other external result could not be undone by the data base management system. Ensuring atomicity of transactions as related to the external (source/sink) data would place severe restrictions on the host language programs and we, like most other data base systems, did not want to impose such restrictions. Internal (data base oriented) atomicity, however, is guaranteed by POREL.

- To simplify Figure 2.2 control lines are not drawn. But they can be deduced from the tasks of the individual processes. We should only mention here that actually the Source/Sink Module (on request from a user) is in charge of starting the transaction execution.

- The Run-time Checker (RTC) will be activated whenever a previously translated and stored transaction is about to be executed. It ensures that all the assumptions made about the data base and the data base schema at translation time (NUA, NOA) still are valid. For example, somebody could have changed the schema description, or the distribution of the relations in the network, or even just the size of relations invalidating the results of

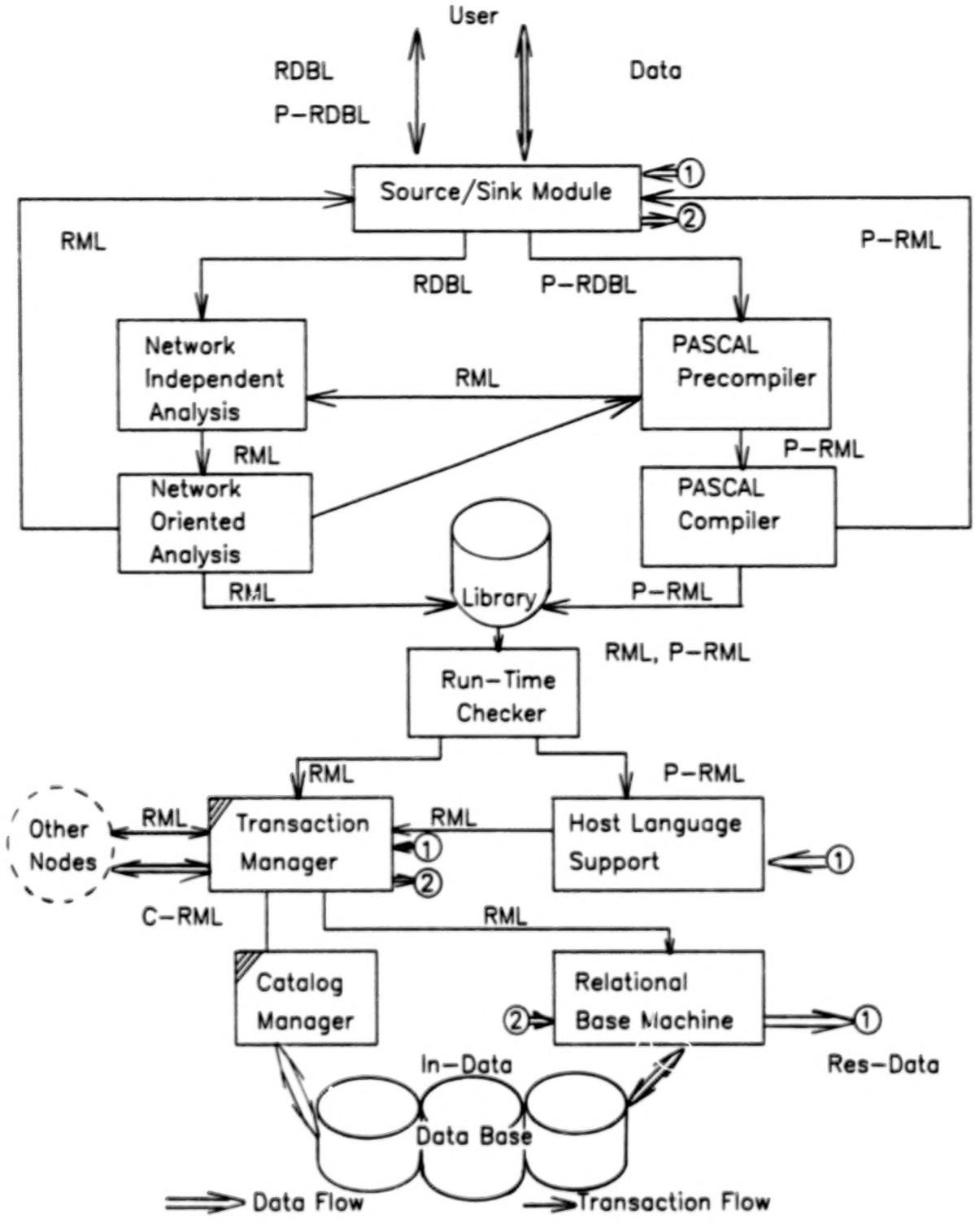


Figure 2.2: The Process Structure of POREL

the subtransaction distribution algorithm. The RTC will start a retranslation of the transaction if the changes in the data base system are such that it can be done successfully; otherwise it will inform the user. Changes in subrelation distribution for example fall into the first category, whereas the deletion of a relation used by the transaction clearly requires user actions to resolve the problem.

- The Host Language Support system controls the communication between the executing PASCAL program and POREL. It sends the RML statements to be executed to the Transaction Manager whenever control in the PASCAL program reaches the corresponding execution point, and it receives the results produced by the RML statements and hands them to the PASCAL program.

- The Catalog Manager is in charge of all accesses to POREL catalogues. All other processes request its services through the use of control commands embedded in RML. Each such command can actually be seen as a small RML transaction specially preprepared for the necessary accesses to the Catalog Manager. That is, the Catalog Manager could be seen as an Abstract Data Type with predefined control operations (C-RML transactions) to be activated by whichever other process requires its services. The Transaction Manager will handle C-RML transactions like any other transaction with the only exception that their processing takes place with higher priority and that they use higher priority features of the Communication System in order to be speedily executed by the system. Ultimately we are thinking to supply a special Catalog Manager where all the information it has to manipulate will be kept in a form which ensures very efficient access. The necessary data structures have already been designed - in many instances interconnected list structures - but not yet implemented. Currently, the Catalog Manager is represented by one instance of the Relational Base Machine. That is, catalog data are kept in relational form and are manipulated exactly like user data, with the only exception that the existence of just a single Catalog Manager on a node eliminates the need for data sharing of catalog data and the predefined RML operations eliminate the need for consistency checking.

  Catalog data are actually organized such that the Network Independent Analysis has all its catalog data always available locally in the network. They are stored in the so-called Short Catalogs, containing the, in general, slowly changing schema description information NUA needs. The information for the Network Oriented Analysis and the other POREL components is kept in the so-called Long Catalogs which may not contain locally all the information required. Since this information e.g. the size of a subrelation, the value distribution in its attributes, the storage and access strategy selected for it, is changing relatively frequently it is only kept at network sites where the corresponding subrelation is actually stored. Using the Transaction Manager - this is one of the reasons why the Transaction Manager is involved in catalog access; recovery is the other - the respective C-RML transactions are sent to the proper network sites and the results are returned to the requesting POREL process. The catalog and catalog accesses have been organized such that a redistribution of information between Long and Short Catalog can be done easily whenever actual performance statistics point to inefficiencies in the current distribution.

Note that catalog information in many instances will be kept redundantly in the network. The C-RML transactions are prepared such that their execution automatically will ensure catalog consistency throughout the network.

Summarizing we can say that POREL is a relational distributed data base management system working on an inhomogeneous computer network and providing the user with a unified view of all the data kept in the system. Through its different external interfaces users of different expertise can work concurrently on POREL.

The minimal system configuration required to run POREL has to contain:

- o a CPU with 64 k byte storage (actually 64 k byte addressing space in a process), where with a single CPU POREL behaves just like a centralized, multi-user relational data base management system,
- o two disk drives to provide for a reliable store
- o some source/sink device, e.g. a terminal
- o an operating system with
  - a. multi-process handling
  - b. local interprocess communication
  - c. some direct access, block oriented file system
  - d. a PASCAL compiler

This concludes our overview of the POREL architecture. The implementation of the prototype has required about 70 person years and is now in the process of being ported to a network of VAX/11 systems.

3. A COMPARISON OF DISTRIBUTED DATA MANAGEMENT SYSTEMS

In this chapter we shall give a brief comparison of POREL with a number of data base management systems currently existing either as prototypes or products or in the late development stages.

SDD-1 of the Computer Corporation of America was the first distributed data base system to be developed. It is oriented towards military applications where an apriori known set of transactions is to be executed. The system works in an environment where communication failures and breakdowns of

nodes may occur more frequently than is usually assumed in the data base world.

SIRIUS/DELTA of INRIA was developed in cooperation with the French industry and it had as one of its objectives the integration of existing data bases into the total system.

R* is currently being developed by IBM Research in San Jose and it is an outgrowth of the (centralized) DBMS prototype System R that eventually led to the IBM Product SQL/DS.

VDN (VERTEILTES DATEN NETZ) was jointly developed by the Technical University of Berlin and Nixdorf Computers. It assumes two back-to-back processors in a node of the network where one processor functions as a software implemented data base engine.

ENCOMPASS of Tandem was developed for Tandem's NONSTOP hardware and its operating system. Strictly speaking it is more of a distributed file system than a data base management system as it misses some of the manipulation facilities commonly associated with data base systems and data base languages. For example it is not location transparent; i.e., a user must know the location of the data in the network and/or the location of the directories containing data partitioning information.

System D was developed by IBM Research in San Jose and was, like ENCOMPASS, oriented toward highly reliable (single failure proof) computing networks. Like ENCOMPASS it uses a record/file type of interface and does not represent a true data base management system. The two last systems have been included here because of their "high reliability" properties since such highly reliable systems are of utmost importance in distributed environments and are somewhat neglected in the other approaches.

Tables 3.1, 3.2 and 3.3 contain a comparison of the systems. As one can see the systems have been developed on a wide variety of computers and communication systems but are usually oriented towards a relational model if not towards the even simplier file/record concept of conventional business oriented programming languages.

	Computers	Communication Network	Network Speed	Data Model	Catalog Strategy
POREL	PDP11, VAX	X.25, Telephone, Ethernet	Low/ High	Relational	Local Long; Global, Replicated Short
SDD−1	DEC 10	ARPANET	Low	Relational	Replicated Global + Fragmented Local + Cache
SIRIUS	Realite' 2000	Ring 19200B	Low	Hetero− geneous + Relational Extensions	3 Local + 3 Replicated Global
R*	370, 3033	SNA/CICS	Low Medium	Relational	1Local + Cache
VDN	Intel 8086, RMX	X.25, Ethernet	Low/ High	Relational, Links	1 Global
ENCOMPASS	NONSTOP	Guardian/Expand	Low/ High	Files & Record	1 Global & Remote Access
SYSTEM−D	Series/1	Insertion Ring	High	Files & Record	1 Local + Relational Managers Data Base

Table 3.1

	Data Distribution	Data Definition & Manipulation Language	Query Processing
POREL	Horizontal by Predicate	RDBL + Pascal Cursor	Compilation, Query Tree Optimization
SDD–1	Horizontal, Vertical	Data Language	Single Command Transaction, Join–Optimization
SIRIUS	Horizontal, Vertical, Indirect by Predicate	Francois (interactive)	Location & Predicate Optimization, Predefinition Strategy
R*	Naming	SQL + PL/1 Cursor	Compilation, Join–Optimization
VDN	Horizontal by Predicate	Single Record, Record set, Link–Chains	Location Predicate Optimization
ENCOMPASS	Horizontal by Key Ranges	Screen COBOL	Location Key Optimization
SYSTEM D	Records on Pages	Single Record	Records on Pages

Table 3.2

	Concurrency	Transaction Atomization	Status
POREL	Distributed Locking by System, Preclaiming	2 Phase commit, Stable Storage Intention Lists	Prototype
SDD–1	Conflict Graph, Time Stamps	2 Phase Commit, Intention Lists, Save Delivery	Military Use Prototype
SIRIUS	Distributed Locking by System, Prevention	2 Phase Commit, Log	Prototype Selling
R*	Distributed Locking by System, Detection	2 Phase Commit, Shadow Page, Log	In Prototype Development
VDN	Distributed Locking by User, Time–out	2 Phase Commit, Different File, Log (after image)	In Product Development
ENCOMPASS	Distributed Locking by User, Time–out	2 Phase Commit, Audit Trail (before & after image)	Product
SYSTEM–D	Distributed Locking by System, Time–out	2 Phase Commit, Log (after image)	Cancelled

Table 3.3

4. CONCLUDING REMARKS

The general introduction into POREL and the short specification of the important properties of a number of other distributed data base management systems show the importance of the relational data model in this world. Distributed data base systems based on hierarchical or network models are bound to be more complicated and without discipline and careful placement of the data they will lead to inefficiencies and high communication overhead.

Many companies are currently working rigorously on distributed operating and data base management systems and we can expect a number of products to appear on the market in the next two to three years. Whether the trend will continue with the relational model or will eventually move into the field of semantic data models (that is, object oriented data bases) is hard to predict, but with knowledge based information systems and with the addition of new data types, as for example, pictorials, 3D solids, voice, text, and drawings, the move away from the simple relational model seems to be unavoidable. A more detailed analysis of this situation is however beyond the scope of the present paper.

5. LITERATURE

[1] ISO/TC97/SC16, Data Processing - Open Systems Interconnection - Basic Reference Model (December 1980).

[2] P.A. Bernstein et al. "Introduction to a System for Distributed Databases (SDD-1)", ACM TODS 5, No. 1 (1980).

[3] Landers, T., Rosenberg, L. "An Overview of MULTIBASE", Distributed Data Bases, (ed. H.J. Schneider), North Holland Publishers (1982).

[4] Chan, A. et al. "Overview of an Ada Compatible Distributed Data Manager", Proceedings 1983 ACM SIGMOD Conference (San Jose, May 23-26, 1983).

[5] Williams, R., Daniels, D., Haas, L., Lapis, G., Lindsay, B., Ng, P., Obermarch, R., Selinger, P., Walker, A., Wilms, P., Yost, R. "R*: An Overview of the Architecture", Proceedings of the International Conference on Database Systems, Jerusalem (June 1982).

[6] Stonebraker, M., Neuhold, E. J. "Concurrency Control and Consistency of Multiple Copies of Data in Distributed INGRES", IEEE Transactions on Software Engineering, SE-5, 3 (May 1975).

[7] Litwin, W. et al. "Sirius Systems for Distributed Data Management", Distributed Data Bases (ed. H.J. Schneider), North Holland Publishing (1982).

[8] Bertino, F., et al. "DATANET/HERMES: Un Sistema Per La Gestione Di Basi Di Dati Distribuite", Annual Conference, A.I.C.A., Pavia, (September 1981).

[9] Neuhold, E.J., Walter, B. "An Overview of the Architecture of the Distributed Data Base System 'POREL'", Distributed Data Bases (ed. H.J. Schneider), North Holland Publishing (1982).

[10] Tsichritzis, D.C., Lochovsky, F.H. "Hierarchical Data-Base Management: A Survey", ACM Comp. Surveys 8:1 (March 1976).

[11] Taylor, R.W., Frank, R.L. "CODASYL Data-Base Management Systems", ACM Comp. Surveys 8:1 (March 1976).

[12] Codd, E.F., "A Relational Model of Data for Large Shared Data Banks", CACM 13:6 (June 1970).

[13] CCITT, "Draft Revised Recommendation X.25, ACM Computer Communication Review (June 1980).

[14] Gabler, H., Tietz, W. "Datenkommunikation in den Fernmeldenetzen der Deutschen Bundespost", Informatik Spektrum 4:1 (February 1981).

[15] Studer, R. "Functional Specification of a Decision Support System", Proceedings of the 5th VLDB Conference, Rio de Janeiro (1979).

[16] Chamberlin, D.D., et al. "SEQUEL 2: A Unified Approach to Data Definition, Manipulation and Control", IBM J. Res. Develop. 20:6 (November 1976).

[17] Schmidt, J.W. "Some High Level Language Constructs for Data of Type Relation"., ACM TODS 2:3 (September 1977).

[18] Walter, I. "POREL Design Specification Part II: RDBL-Language Description", Technical Report 78/5, IFI, University of Stuttgart (1978 (in German)).

[19] Bohme, K. "A Communication Service Applied in a Distributed Data Base System", Proceedings 3rd ECI-Conference, Lecture Notes in Computer Science 123, Springer Verlag (1981).

[20] Bohme, K. "The Transport Service of an Interprocess Communication System", Proceedings HICSS-15, Honolulu (1982).

CONTROL DATA ICEM - A VENDOR'S IPAD-LIKE SYSTEM

Harley D. Feldman
Control Data Corporation

Integration has been the key to the IPAD program since its inception. IPAD's goal was to integrate aerospace applications used in support of the engineering design process. It is still the key goal, now having evolved to a design centered around the use of data base management, networking, and global user executive technology.

Control Data Corporation has been developing an integrated CAD/CAM system modeled in part after the IPAD program and containing elements of the program's goals. The total program is called ICEM. Integrated Computer-Aided Engineering and Manufacturing. The ICEM program started with the acquisition of AD-2000 from Manufacturing and Consulting Services in 1977 and Synthavision from Mathematical Analysis Group, Incorporated, in 1978. AD-2000 has evolved over the years to a production geometry creation and drafting system now called CD/2000. Synthavision has grown to be a full-scale 3-dimensional modeling system, the ICEM Modeler.

Shortly after the acquisition and enhancement of these software systems, it became obvious that information common to them would be utilized by the design engineer. This was especially true of part geometry. About that same time, the National Bureau of Standards developed the Initial Graphics Exchange Specification (IGES), allowing different CAD/CAM systems to share geometry definition upon adhering to the IGES neutral file format. It was natural at Control Data to develop software to exchange geometry between CD/2000 and other CAD/CAM systems, allowing our customers some level of integration among dissimilar systems. Since that time CD/2000 and the ICEM Modeler have been connected through IGES files, allowing further integration of the design process.

The problems then confronting the CDC CAD/CAM software designers were 1) a never-ending connection of more applications to share common information (in addition to geometry) and 2) the management and control of the data rapidly being created by the software packages. Also as more capabilities were given to the CAD/CAM system user, the use of the system was becoming more complicated and diverse. Looking at the IPAD program, the concepts of data management and executive user interface appeared to solve these problems. The solution developed at Control Data to model the IPAD ideals was called the Engineering Data Library (EDL).

EDL has four major components, 1) the user interface to all systems, 2) the interface to the operating system, 3) a file manager for all files under design control, and 4) the design discipline data base. The user logs into EDL and executes his design processes completely in the engineering environment created by the EDL menus. EDL determines valid states of the design process and constrains the user to perform in that environment. EDL accesses all of the necessary application software and appropriate files for the user to perform his design functions.

This level of integration provides an IPAD look-alike environment in the sense that it appears to the user that his processes, applications, and data are all integrated into one common system as was desired by the IPAD design. Unfortunately, while the appearance is good, the true integration of data elements in a common data base has not been achieved.

Work is proceeding in this area. Since the geometry is such a fundamental data element that is shared in various disciplines through the design process, it has been the first candidate for true integration in a common data base. A methodology used and marketed by Control Data in the design of information systems, Information Analysis, has been utilized to understand the problem of managing common geometry. Information Analysis has shown the description of a relational data base structure to support many of the common geometry modeling techniques from solid objects and surfaces on down to points, lines, and arcs. The IPAD staff has independently generated similar data structures for the geometry primitives chosen by them. Both analyses point to the fact that current data base management system technology can be used to support these data structures.

However the bad news which comes along with this analysis is that the applications utilizing this geometry will have to be modified to use this definition in the data base. If that were not bad enough, each application of user view of the geometry is probably unique, requiring different transformations of the data to and from the data base. This is the area where the most research and eventually development work needs to be done. IPAD has begun some of this work by pre- and post-processing geometry between IPIP and the AD-2000 CAD/CAM system. Control Data has demonstrated this work between CD/2000 and the Information Management Facility (IMF), a three-schema data base management system.

Since investigating geometry data management, IPAD has discovered the notion of structured data processing. This capability offers great promise in the ease of accessing geometry information from future engineering data bases. Control Data is investigating the usage of this technology in future data base management system capabilities.

IPAD has been a role model for the ICEM system developed by Control Data. Many of the concepts were seeded in the IPAD program and carried forward into CDC products. Through future IPAD research, it is hoped that further discoveries will be achieved leading to technology supporting the truly integrated Engineering and Manufacturing processes.

FUTURE DEVELOPMENTS OF A HYBRID CAD/CAM SYSTEM

J. Z. Gingerich
Sperry Corporation
Applications Development Center
Blue Bell, PA

ABSTRACT

The challenge of the 80's for CAD/CAM systems is to intergate proven CAD/CAM techniques with the system advances of this decade. The Hybrid CAD/CAM System (UNIS*CAD) concepts and future developments address this issue of integration.

INTRODUCTION

The current CAD/CAM systems have evolved in the past ten years into useful and productive industrial tools. These surface modeling systems started with the basic curve and point capabilities (wire frame) for drafting. For the most part, they have further integrated manufacturing and analysis applications and have provided surface capabilities within the original drafting system core. (The core is defined by the support interfaces to the actual application software. Examples of the core are: the data base structure and utilities, the graphic display utilities, and the computer system interface.)

At the other end of the spectrum, universities and some industrial and commercial concerns have been developing volumetric modelers with strengths of a robust volumetric data structure providing exceptional model analysis and display. These systems generally lack the productive user interfaces and applications of the current systems and use far greater computer resources. In all systems, non-graphic data handling is limited and the integration of non-graphic or graphic data into a corporate data base or network is almost non-existent.

Recently, work has been done to merge the existing surface systems with the emerging volumetric modelers in order to provide the best of both approaches: a robust volumetric data structure, productive user interface, and integrated applications. Most of this work to date is involved with trying to fit one system into the core of the other. Additionally, this work has not considered the integration to corporate data base or the benefits of the emerging hardware and software technologies such as relational data base machine, distributed architecture, and intelligent 3-dimensional graphic displays. Consequently, most of the work to date is considered evolutionary in that one system will evolve with the apparent capabilities of the other. However, the core of the one system not being designed to naturally accommodate the other is limited and probably will not address the challenge of the 80's.

The Hybrid Definition

The definition of hybrid is "the offspring of genetically dissimilar parents or stock".[1] Hybridization is not evolutionary, but rather a revolutionary process of creation. This paper presents the concepts of a Hybrid CAD/CAM System that considers the facets of the "dissimilar parents" by providing a revolutionary design of the system core, rather than fitting one into the other. It also presents the applications and user interfaces facilitated by the system concepts.

System Overview

As just stated, the Hybrid System concept provides the revolutionary core for the offspring of the hybridization attempt. (Keep in mind that the offspring has the desirable qualities of the current CAD/CAM surface based systems and the developing volumetric modeling system, plus being able to embrace the emerging technologies of the 80's.)

The core is defined as the supporting interfaces to the CAD/CAM applications software, specifically the Graphics (human) Interface, the Data Interface, and the Computer System Interface. Figure 1 illustrates (very abstractly) the structure of the Hybrid System core and the relationships of the various applications and system hardware. The outermost cylinder of the core does not change between devices or systems, but rather provides an invariant interface for all the applications at a relatively high level of functionality. This independence is necessary to allow the Hybrid System to embrace the new technologies of the 80's. The innermost cylinders of the core are device and system specific. Changes and additions are accommodated here and are virtually transparent to the CAD/CAM applications.

The remainder of this paper presents more detail about the Hybrid System's core (the Graphic Interface, the Data Interface, and the Computer Systems Interface), presents the Part Descriptor (Design) application which unifies the two dissimilar parents; the surface and volumetric modelers, and presents other applications and user facilities available and planned for the UNIS*CAD, Hybrid CAD/CAM System.

THE GRAPHICS (HUMAN) INTERFACE

The Hybrid System's core Graphics Interface is designed to provide not only an independent display and hard copy device interface, but also a higher level of facilities for human interaction. All the applications of the Hybrid System use these facilities which provide a consistent ergonomically designed graphics window for the user. Contained in the Graphics Interface are all the facilities for menu handling and presentation, the applications' graphics output, the interactive display transformations, the message handling, and the graphic selection capabilities. Since all the applications use these facilities, a consistent style between the applications is unavoidable. All graphics operations are identically presented and controlled between applications.

The Display Format

Figure 2 illustrates the display format used for all the applications. The display Work Area contains the application's output and is the entire displayable area of the display. The Dynamic Menu Area is the portion of the display where the user is presented and selects the various menus to control the application's operations. Also, in this area, verification of user input is displayed when supplying non-graphic data. The immediate Menu Area contains the constantly available general operations all the applications use. Examples of these operations are Help, to request detailed information about the Dynamic Menu operation selected; Display, to control the viewing operations selected, and display, to control the veiwing operations of The application's output such as dynamic rotation, panning, and zooming. the Message Area contains the application's messages to the user. These messages give constant feedback as to where the user is in the applications, what is required to proceed, and, if user or system error occurs, what went wrong and how to resolve it.

Graphics Input

The graphics input facilities provided by the Hybrid System's Graphic Interface emphasize the desirability of retaining the user's attention on the screen at all times. To achieve this, the Graphics Interface supports one primary input device (the pen and tablet) to control all input operations. This includes menu and graphic element election, the control of the dynamic viewing transformations, and the input of typed information. This primary support eliminates the user's having to look away from the display to use a keyboard of function keys, for example. Secondary devices are supported but are not required to operate the applications.

The Menus

The menu facilities provided by the Graphics Interface are designed to be operational in their use, not procedural as most CAD/CAM systems tend to be. A procedural menu is one where the user selects from a list of fixed steps commands. An example of a procedural menu for Line Creation follows:

```
LINE
   SCREEN POSITION
   KEY-IN COORDINATES
   TANGENT TO TWO CURVES
   THRU POINT AND HORIZ OR VERTICAL
   THRU POINT AND TANGENT TO A CURVE
   PARALLEL TO A LINE, TAN TO A CURVE
      etc.
```

The Hybrid System's operational menu is designed to allow the user to select various operators in the menu that compose commands. An example of the Hybrid System's operational menu for Line Creation follows:

```
CREATE LINE
   SCREEN POSITION
   KEY-IN
   INTERSECTION OF ELEMENTS
   ON ELEMENT
      NORMAL/PERPENDICULAR
      TANGENT
      AT ANGLE
      PROJECTED
   BETWEEN ELEMENTS
      MULTIPLE
   FROM ELEMENT
```

The use of the operational menu is to select geometric elements on the display Work Area combined with the operational menu operators, thereby composing a procedure to create a line.

There are two important capabilities an operational menu technique provides. First: flexibility. The user composes the commands using operators in the order and function that defines the design intent exactly. (For example: the operational line creation menu with the 11 operators provides 1300 combinations of operators and selected geometric elements to produce a valid line element in the Part Descriptor application. A procedural menu would require over 100 separate menu items to attain this flexibility). Second: capture of exact design intent at creation time. Because the operational menu specifically relates selected elements to functional operators, the design intent is exactly defined and may be captured as positional relationships (presented later in this paper). This design intent is then available for communication to the other CAD/CAM applications in a non-ambiguous form.

The Graphics Interface provides the above facilities for the applications in a high level, device independent environment. This ensures consistency in the user's graphics window of the applications and allows the Hybrid System to embrace new display and ergonomic advances as they become available.

THE DATA INTERFACE

Much thought has been given to a true integration of CAD/CAM applications. Most of the work in this area is focused on the creation, storage, and manipulation of data. The Hybrid System design also focuses on the Data Interface facilities as the key to the effective integration of CAD/CAM applications.

The Multi-Schema Approach

The IPAD (Integrated Programs for Aerospace-Vehicle Design) Project[2] has done considerable work in the area of data integration for the aerospace manufacturing industry segment. The project has taken the three schema definition of data structure[3] and expanded it to address heterogeneous computer networks. The multi-schema definition is the result. The Hybrid System accommodates a heterogeneous computer network of remote and host computer systems as illustrated in Figure 1. It is necessary to accommodate not only a distributed system architecture for computing in the Hybrid System but also to distribute the data (and not lose the integration). The multi-schema approach effectively allows this.

The multi-schema structure for the Hybrid System is illustrated in Figure 3. Three classes of schemas are represented: the external schemas (application views), the conceptual schema (comprehensive corporate definition), and the internal schemas (internal computer representation). The external and internal schemas are mapped or transformed from the conceptual schema as illustrated. The computer dependency is only in the internal schemas and their mapping of the conceptual schema.

The Hybrid System Conceptual CAD/CAM Schema

The Hybrid System provides a conceptual schema facility in the form of a host computer integrated data base facility. This facility's flexibility for query, update, and manipulation allows the mapping to the external and internal schemas and provides an interface and/or facility for the entire corporate data. The external schemas in the Hybrid System are mapped from the conceptual schema into either remote or host computer network (CODASYL)) data bases depending on the application needs.

The design and implementation of the conceptual schema is provided by the Hybrid System for the data needed by the mechanical manufacturing industry CAD/CAM applications. This conceptual CAD/CAM schema is composed of four data types that together conceptually define the mechanical manufacturing industries' product. The four data types are illustrated in Figure 4. These data types are created, manipulated, and retained by using the Hybrid System's Part Descriptor (Design) application (presented later in the paper).

The Pure Element Data Type. The pure element data is the geometric data necessary to describe a unique element. This canonical data takes the current standards activities such as the American National Standards Institute's Y14.26 committee and the National Bureau of Standards Initial Graphics Exchange Specifications (IGES) as an indication of the data content required in the pure element data.

The Topologic Relationships Type. The topologic relationships are best served by a representational structure similar to Baumgart's winged edge representation.[4] This structure is selected due to its ability to access perimeter information in a quick, orderly manner and due to its ability to provide material and non-material side information. It is also the structure that almost all of the current volumetric modeling activities have adopted.

The Positional Relationships Type. Not to be confused with or thought redundant to topologic relationships, positional relationships describe a Part Descriptor element's position in space, relative to other elements. Dependent on the element chosen, the user supplies positioning information that should be maintained due to its functional importance.

Figure 5 illustrates two functionally unique descriptions of a slot. The left example has the vertical faces of the slot, F2 and F3, both positioned relative to face F1. The right example, however, has only the one slot face, F2, positioned relative to F1. Even though both descriptions generate the same nominal slot geometry, the design intent of the slot and the manufacturing implications for the two descriptions are completely different. Additionally, any changes to the dimension B would affect the two descriptions differently. The slot is of variant width in the left example, invariant in the right.

The Attribute Data Type. The schema must accommodate a facility to allow non-geometric (part generic) information to be associated with the geometric element data. The ability to create and manipulate attributes for any of the Part Descriptor elements is provided. The creation and manipulation of attributes can be done while creating the elements or at a later time. The attributes have a name and value association and also a practically unlimited number of subattributes to an unlimited number of levels as shown:

```
Part Descriptor Element

    Attribute 1 Name.............Attribute m Name

    Attribute 1 Value

        Subattribute level 1.1 name.......level 1.m
        Subattribute level 1.1 value

                    .
                    .
                    .
        Subattribute level n.1 name.......level n.m
        Subattribute level n.1 value
```

Nominal vs. Variational Geometry

It is not acceptable to accommodate nominal geometric descriptions only. Since the geometry will eventually be manufactured, and a nominal geometry can not be manufactured, the Hybrid System must deal with variational (toleranced) geometry. In order to do so, both conventional dimensioning and tolerancing and modern dimensioning and tolerancing considerations are provided in the Hybrid System's CAD/CAM conceptual schema.

Conventional. Positional relationships between elements are represented by distance parameters (dimensions) and a range of allowable variables (tolerances) to them. By allowing variable parameters in the positional relations, the Hybrid System accommodates conventional methods of variable geometric description.

Modern. Allowable variations specified with respect to the object's function (feature constraints). Feature constraints are of two types: intrinsic and extrinsic.[5,6]

Instrinsic: These are feature control statements that deal solely with the unique element (faces generally). Examples are roundness, cylindricity, flatness and straightness. These feature constraints are represented as attributes to the Part Descriptor's elements.

Extrinsic: Extrinsic constraints are based on parameters to Part Descriptor elements other than the element being constrained. Examples are parallelism angularity, perpendicularity, profile, runout, concentricity, and true position. Here again, the constraints are handled as attributes associated with Part Descriptor elements (faces).

Datums are the 'targets' (assumed to be exact) for the establishment of extrinsic constraints. Datums may be defined as attributes of vertices or edges, but mostly are attributes of faces. The user needs to define the name datums as a part of the variational geometric description process.

The modern tolerancing of instrinsic and extrinsic constraints and datums are all handled in the Hybrid System as special program identifiable Part Descriptor element attributes.

THE COMPUTER SYSTEM INTERFACE

The Hybrid System Computer System Interface provides a high level, device independent facility for the applications. It allows the use of an architecturally heterogeneous computer network and provides an easy way to upgrade that network as computer systems advance. This interface handles computer to computer and computer to peripheral control and communications. This interface forms the innermost cylinder of the Hybrid System core. Figure 6 illustrates an arbitrary slice through an application of the Hybrid System cylinder. Note the relations of the application to the core as it moves from the device and system independent applications towards the dependent core.

THE PART DESCRIPTOR (DESIGN) APPLICATION

The Hybrid System's Part Descriptor (Design) application is a unique application in that it generates data for the CAD/CAM conceptual schema where the other applications primarily use a view of that data to generate their own application data (such as a drawing file, CLDATA statements, or finite element model). The Part Descriptor is not unique in it's use of the Hybrid System's core, however. It too uses the interface provided by the core and consequently maintains the same appearance and flexibility of the other applications.

The following geometric elements are provided by the Hybrid System Part Descriptor application:

- ASSEMBLY
- PART
- SOLID
  - BLOCK
  - CYLINDER
  - CONE
  - SPHERE
  - EXTRUSION
  - REVOLUTION
- FACE
  - PLANE
  - REVOLUTION
    - Cylindrical
    - Conical
    - Spherical
    - Toroidal
    - General
  - RULED
    - Developable
    - General
  - TABULATED
  - FILLET
  - SYNTHETIC
- EDGE
  - LINE
  - ARC/CIRCLE
  - CONIC
    - Ellipse
    - Hyperbola
    - Parabola
    - Five Condition
  - SYNTHETIC
    - Planar
    - Space Curve
- VERTEX

The hybridization of the dissimilar modeling techniques is evident in the Part Descriptor. Here the user may use volumetric or surface modeling in a unified application. The surface (and wire frame) modeling is a part of the Part Descriptor's natural capabilities. The volumetric modeling is an integrated extension of the Part Descriptor's capabilities. The volumetric modeling capabilities are based on the Baustein Geometrie (or COMPAC) system that was originally developed at the Institute for Production Systems and Design Technology (IPK), Berlin. It provides for solid constructive geometry modeling with Boolean operators and primitive solid shapes of blocks, cylinders, cones, spheres, volumes of extrusion, and volumes of revolution. The primary representation scheme is the boundary evaluation representation.

There are limitations with combining volumetric and surface modeling, however. While volumetric modeling provides a robust data structure and a very convenient modeling technique, volumetric modeling cannot accommodate the more complex, non-analytic surfaces. The surface modeling on the other hand can accommodate the sculptured surfaces, but does not contain the topologic data necessary for volumetric modeling. Therefore, the use of the Hybrid System's Part Description application allows the user to do volumetric modeling with the SOLID elements listed above. At the point of part description where the above FACE or EDGE elements are required, the user may surface model the results of the volumetric boundary evaluation but does so with the understanding that further boundary evaluation is not possible for that part or assembly. This does meet the goals of the of the hybridization though, because the product is completely described by the use of one or both modeling techniques within a unified system. And the resultant model is later used by the Hybrid System applications.

Naturally, as the state-of-the-art advances in the surface capabilities of volumetric modeling, the Hybrid System's Part Descriptor has the proper core for embracing that emerging technology.

THE DRAFTING APPLICATION

The Hybrid System's Drafting application recognizes that Drafting is separate from Design. As Figure 7 shows, the Drafting applications use the Part Descriptor model as the data from which Drafting projections are derived. This separation of the Part Descriptor model and the Drafting data guarantees the control of the geometry model (or master parts data) because the applications are working with views of the master data, not the master data itself. This insures that there is only one master data and that the applications themselves do not change that master data.

The Drafting application provides the facilities to create the views of the desired parts and/or assemblies and project them onto the drawing. Additional two-dimensional geometry and symbols, dimensions, and text can be added to the projections.

THE MANUFACTURING APPLICATION

The Hybrid System's Manufacturing application provides a facility for generating, executing, and verifying APT-1100 programs. Just like Drafting and other applications in the Hybrid System, the Manufacturing application starts with a view of the Part Descriptor master part and/or assembly data in order to generate APT geometry statements. The selected geometries are then referenced for part programming to generate APT tool motion and set-up statements. The APT source statements may then be visually checked and consequently executed by the APT-1100 processor. The CLDATA is then generated and graphically reviewed before post-processing to the N/C machine for manufacturing.

User Interfaces

The facilities for User interfaces to the Hybrid System include:

Data Exchange Windows
User Defined Macros
User Defined Menus
- Text Modification
- User Macro Menu

Graphics Facilities
- Part Descriptor Application
- Drafting Application

General Purpose Computing Facilities
- 1100 System Support
- Standard 1100 DBMS Systems

These facilities, as shown in Figure 8, provide a variety of flexible interfaces to the UNIS*CAD Hybrid System that do not require a user programming knowledge of the Hybrid System itself. Using these facilities, the user can integrate company specific programs and/or systems into the Hybrid System.

Data Exchange Windows. The data exchange windows are well defined, high level FORTRAN utilities to define, create, and retrieve data in the Hybrid System's conceptual schema. The data exchange window, for example, can support the exchange of geometry in the IGES format, can be used as a data base facility for user written applications, or can be an exchange window for user application data base into the Hybrid System's conceptual schema.

User Defined Macros. The user defined macros are repeatable user defined application sessions created and repeated to accommodate family of parts in the Part Descriptor or Drafting application for example.

The user macro facility is available for all the Hybrid System's applications. Again, the macro facility will be consistent in operation between the applications because it is built upon the core of the Hybrid System.

User Defined Menus. The user defined menus allow two facilities: text modification and a user macro menu. The text modification allows the user to change the actual text of the Hybrid System's application menus, help messages, prompt and error messages. This accommodates multiple languages and special terminology requirements of the user's environment. Changing the text of the menus does not change the operation or intent of the menus however.

The user macro menu is a user defined set of menus that presents and executes the user's predefined macros within the Hybrid System's applications.

Graphic Facilities. The user may, by using the data exchange window, get data into either the Part Descriptor or Drafting application thereby being able to have user data graphically displayed and manipulated using the comprehensive two and three-dimensional graphics facilities of the Hybrid System's Drafting and Part Descriptor applications. An example of this use would be to take lofting data generated by the user's proprietary apppli-cation system and viewing that data in the Part Descriptor. Modifications and additions could be done by the Part Descriptor and the resultant model could be used by the Drafting application to generate a mechanical drawing of the lofting geometry. Also, the Manufacturing application could generate the machining information to manufacture the realization of the lofting data.

General Purpose Computing Facilities. Because the Hybrid System's archi-tecture includes a host computer facility, all the general purpose resources of a mainframe processor (or network of processors) is available for user developed applications. And by using the user facilities of the Hybrid System, a true integration of UNIS*CAD and a user's computing processes can be achieved.

SUMMARY

The Hybrid System (UNIS*CAD) is in continuous development and support by Applications Development Center of Sperry Corporation. The Hybrid System's core design addresses the challenge of the 80's: to integrate the proven CAD/CAM techniques of the 70's with the emerging technologies of the 80's.

ACKNOWLEDGEMENTS

The author would like to recognize the following constributors:
M. Carroll, E. Chelius, W. Drean, L. P. Kuan, J. Perneski, and R. Tickell; all of the Worldwide Applications Development Center, Sperry Corporation, Blue Bell, PA, and D. Bickel of the ETW, Scientific Programming Products, Sperry Univac, Sulzbach, West Germany.

REFERENCES

1. The American Heritage Dictionary of the English Language, Houghton Mifflin Company, 1978.

2. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA Conference Publication 2143, Denver, Colorado, 1980.

3. Date, C. J., An Introduction to Data Base Systems, Addison Wesley, 1980.

4. Baumgart, B. G., "Polyhedron Representation for Computer Vision", NCC 1975, AFIPS Press, 1975.

5. Requicha, A. A. G., "Part and Assembly Description Language I: Dimensioning and Tolerancing", TM-19, Production Automation Project, University of Rochester, Rochester, New York, 1977.

6. Gjovaag, I., "Geometric Modeller Project Requirements Analysis", Internal document, Tektronix, Inc., Beaverton, Oregon, 1978.

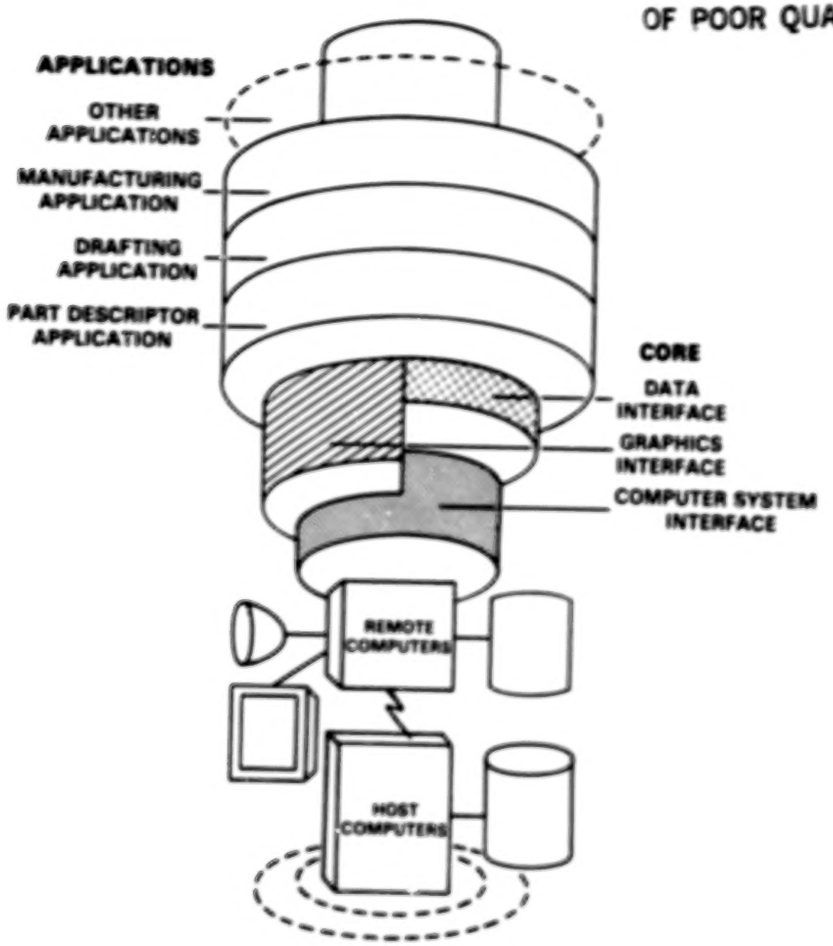


Figure 1.- System organization.

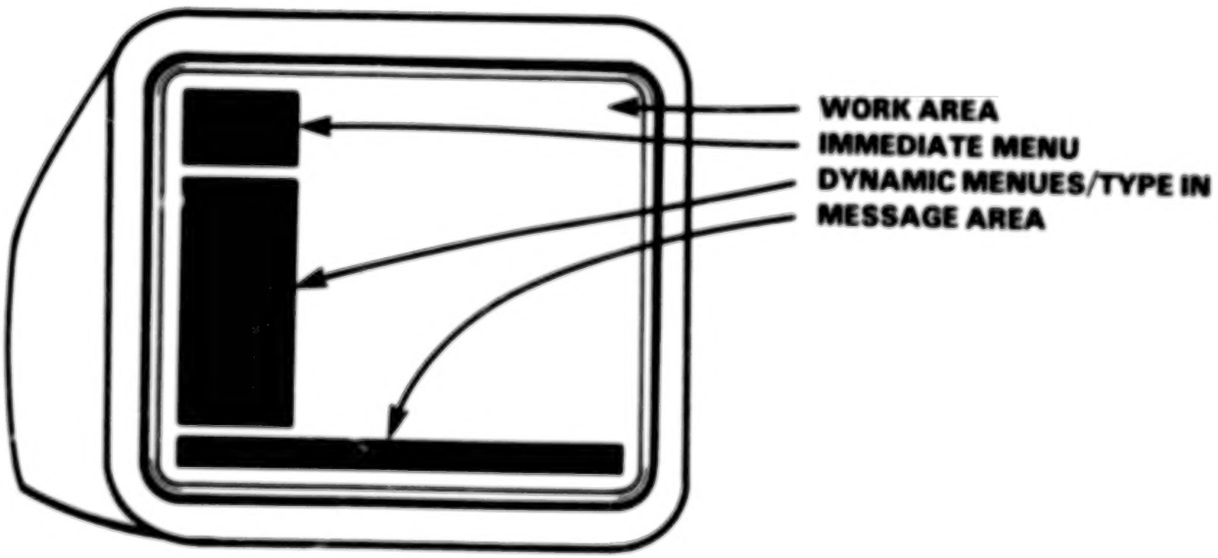


Figure 2.- Display format.

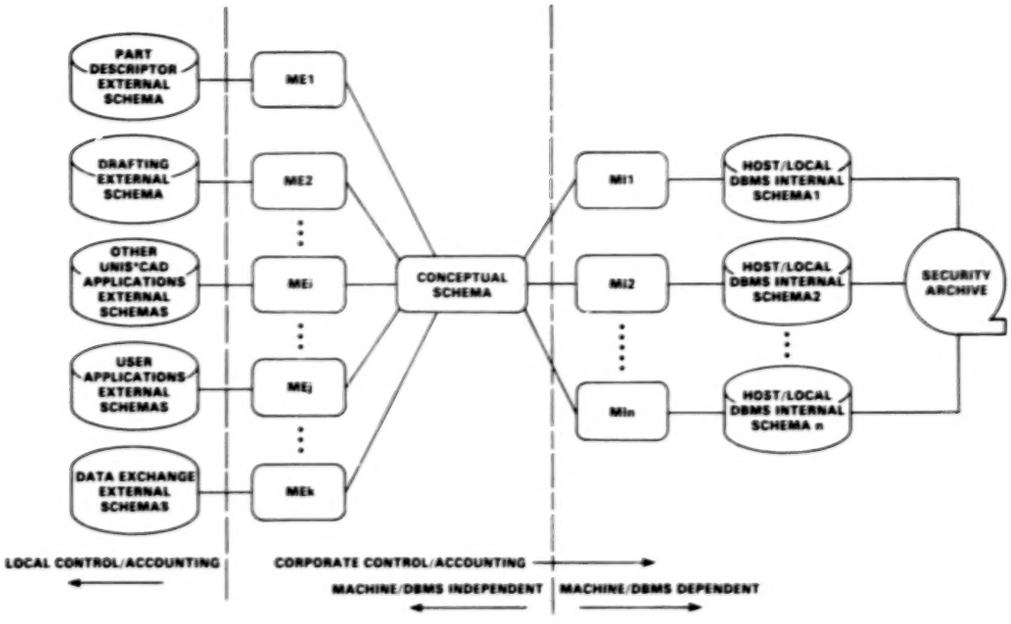


Figure 3.- Multi-schema structure for Hybrid System.

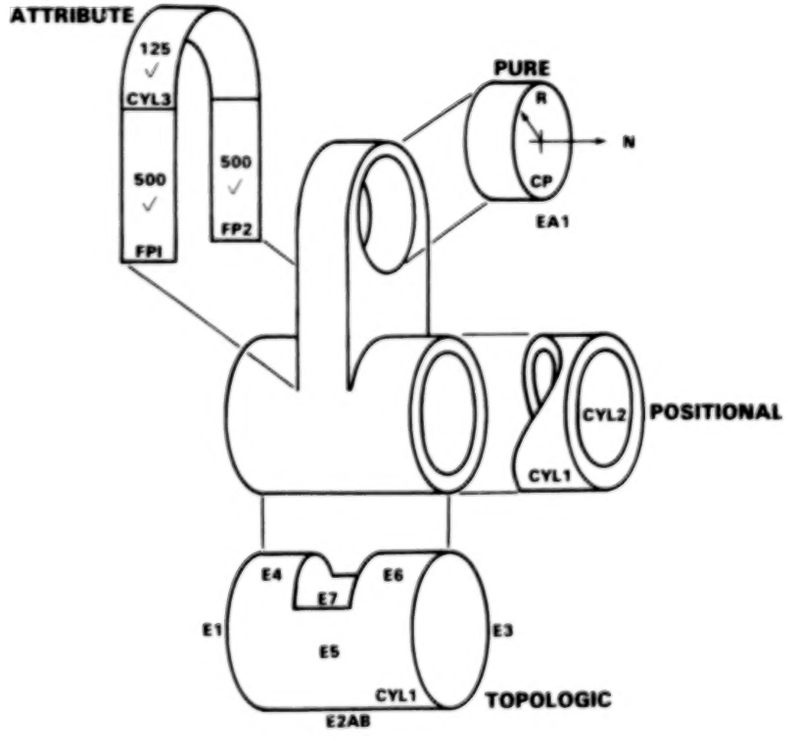


Figure 4.- Conceptual schema data types.

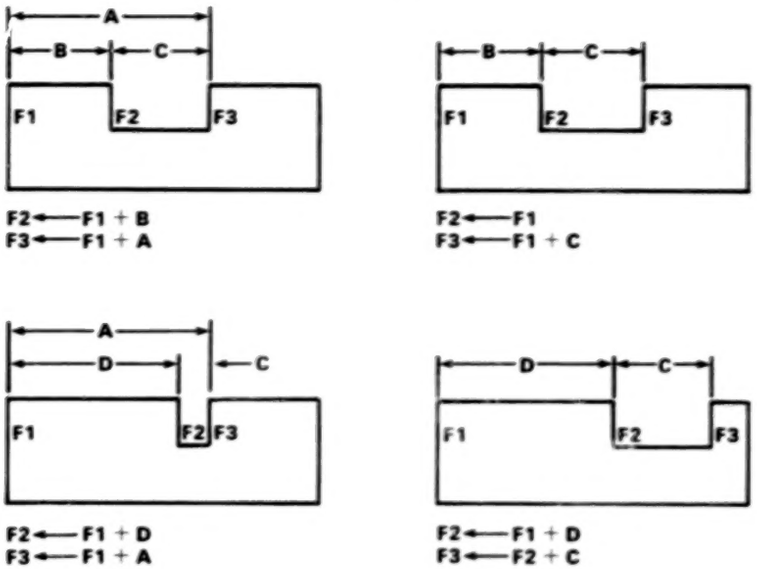


Figure 5.- Positional relationships.

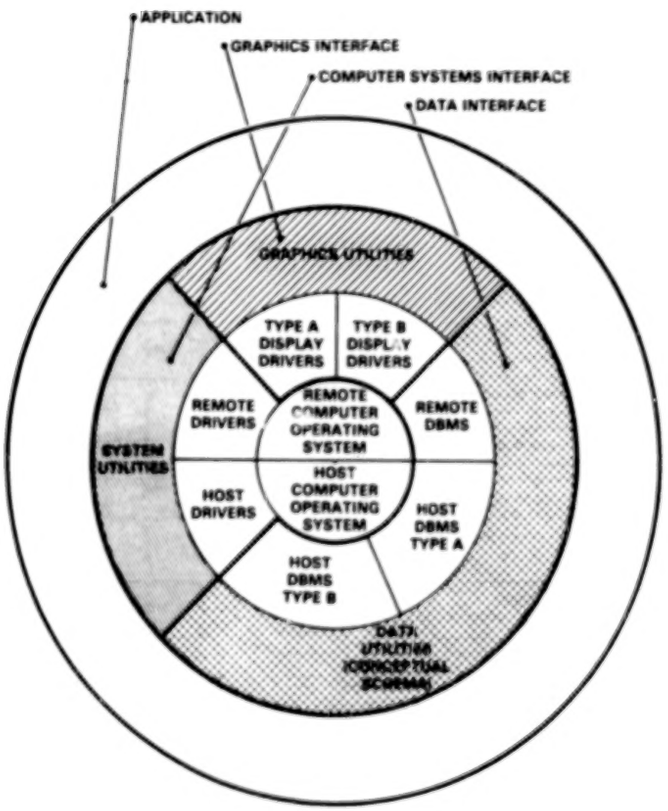


Figure 6.- Slice through Hybrid System cylinder.

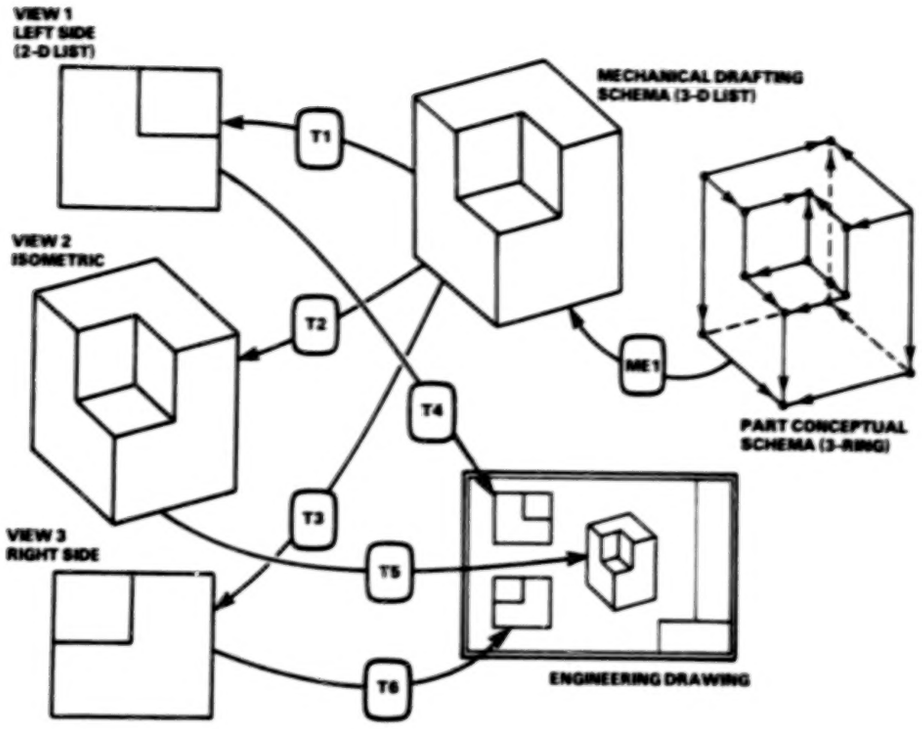


Figure 7.- Drafting view projections.

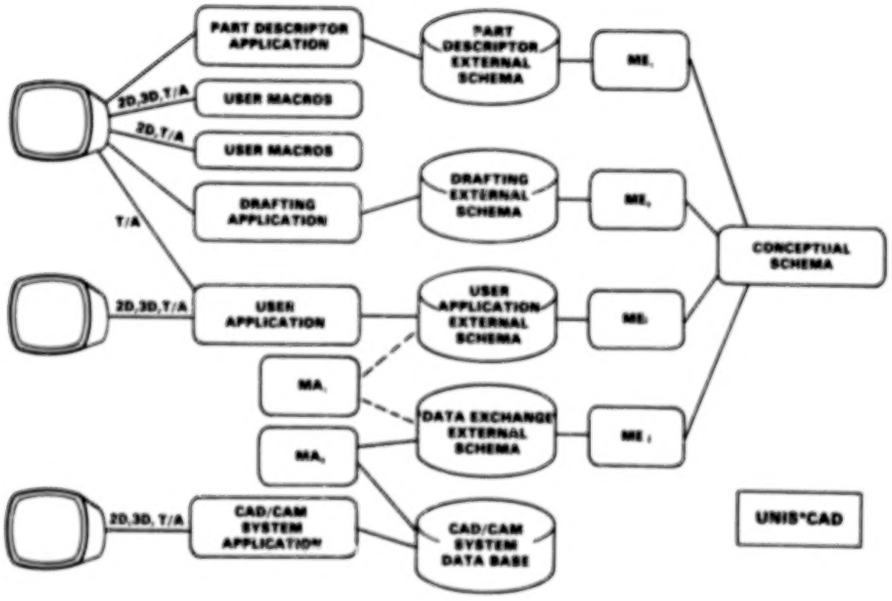


Figure 8.- User interfaces.

RIM AS AN IMPLEMENTATION TOOL FOR A DISTRIBUTED HETEROGENEOUS DATABASE

Yuri J. Breitbart
Larry R. Hartweg
Amoco Production Company
Tulsa Research Center

SUMMARY

The Amoco Production Company and Standard Oil Company (Indiana) computing environment encompasses a highly Distributed network of Heterogeneous Database Management Systems (DHDBMS) and query languages, such as IBM's IMS, SQL and DB2, ASI Inquiry, Infodata's Inquire, Information Builders' FOCUS, and a variety of access methods, file structures, and network protocols including Boeing Computer Services' RIM. The operating system environments for these database managers include IBM MVS, IBM VM, IBM XT/370 Personal Computers, Perkin Elmer, and S.E.L. Interactive systems have long been available for accessing individual databases, but new user queries involving DHDBMS have generally required significant development efforts and the creation of multiple batch processing interfaces.

An Amoco-enhanced version of BCS/RIM is currently being used as an effective tool in the implementation of systems which simplify the development of new distributed database applications. "Another Distributed Database System" (ADDS) is an ongoing Amoco Computing Research project to investigate the feasibility and add to the knowledge of effective DHDBMS. Although many Amoco database applications are using RIM as a homogeneous DBMS (see Appendix), a primary goal of ADDS is to avoid the re-design of hundreds of man-years of existing DBMS-specific applications.

The ADDS prototype now supports interactive, ad hoc retrieval from several of the Amoco/Standard DHDBMS. We will outline the ADDS conceptual design, the usage of RIM in several components of ADDS, and some enhancements to RIM that were used by the developers of the ADDS prototype. Topics covered by this paper include:

I. ADDS Overview

II. Composite Database Dictionary/Directory

III. User Interface and User Profiles

IV. Subrequest Execution

V. Merger/Formatter

VI. A Transportable Implementation

I. ADDS Overview

ADDS presents the user with a relational view of composite databases (CDBs). The physical database (PDB) components which make up a CDB may be geographically distributed on heterogeneous DBMSs running under a variety of computer hardware and operating systems. The various network protocols which link the DBMSs may also be dissimilar. An ADDS user may request data from a CDB without understanding which PDBs are involved or how the data is distributed or structured within the PDBs. The ADDS user's logical view of the composite data is a single two-dimensional relation. The ADDS query language is similar to the RIM and SQL relational query languages. Any combination of fields from a CDB may be requested. The resulting rows may be restricted (with a WHERE clause) and sorted in ascending or descending sequence based on the values found in any combination of fields.

Figure 1. Major components of the ADDS conceptual architecture

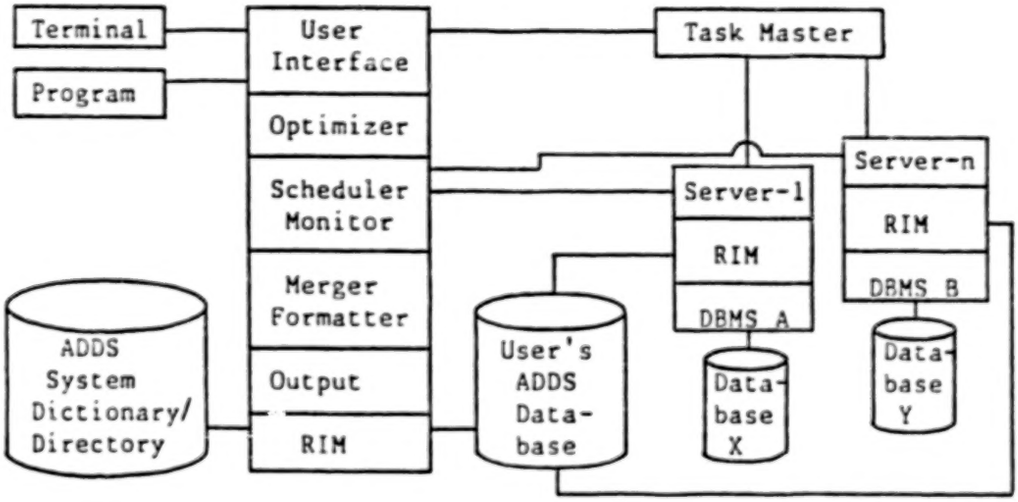


Figure 1 depicts the conceptual architecture of ADDS. Once a query has been entered through the user interface (from a human or a program), ADDS must access its dictionary/directory to verify the request, determine an optimum method of scheduling, obtain parallel subrequest processors from the task master, schedule the subrequests, and merge, format, and distribute the output. One objective of this approach is to insulate end users from the complexity of the distributed composite database structure and the underlying network and DBMS commands and query languages. The task master/server concept allows new data managers and network protocols to be rapidly supported by adding new servers. A server may communicate with a local DBMS or remote computing systems, including a remote ADDS system. The number of parallel task processors and the number of servers per data source are tuning parameters which may be changed while ADDS is running.

The ADDS prototype research project is using RIM to speed implementation and to construct a transportable system. RIM is incorporated in the design of the ADDS user interface, the directory, and the intermediate output from the servers. RIM is also used for passing restrictive qualifications to dependent servers ("semi-join" operation) and for performing multi-server, merge-and-output formatting processes. The following discussion will be limited to those portions of ADDS which use RIM as an implementation tool.

II. Composite Database Dictionary/Directory

The primary purpose of the ADDS directory is to provide the information required to translate queries from a user's view of a composite database into a set of subrequests to the component data sources. Secondary directory information for optimizing, scheduling, merging, formatting and output will be discussed later. Logical database (LDB) definitions are used to assign consistent field names and formats to similar fields in different PDBs that were created independently, without consideration for combining their output with other PDBs. When a new CDB definition is to be added to the system, a database administrator describes the set of LDBs in the directory RIM database. Each LDB name within a CDB is represented by a single row in the "Component" relation.

Amoco enhancements to RIM support terminal-dependent, full-screen database display and editing, which simplify the ADDS database administration (DBA) function. The relationship between logical field names and PDBs and the physical/logical mapping is described by rows in the "Fields" relation of the ADDS directory. To show key points, we will introduce the following example CDB (Figure 2) and refer to it throughout this paper. (The relationship between the size of LDB1 and LDB2 will become important in our discussion of the optimizer.) For simplicity, only the most significant dictionary/directory field values are shown in our example.

Figure 2. Example "WELLS" Composite Database Definition

LDB1 - A local RIM database with one megabyte of data:
LDB1 Fields: WELLNUM, DEPTH, POROS, FORMAT

LDB2 - A remote IMS database with sixteen gigabytes of data:
LDB2 Fields: WELLNUM, DEPTH, PERM, WELLNAME

The component relation in the directory would contain:

CDB	PASSWORD	LDB	DBMS	DESCRIPTION	etc.
WELLS	xxx	LDB1	RIM	Local wells database with one megabyte	...
WELLS	xxx	LDB2	IMS	Remote wells database with 16 gigabytes	...

The fields relation in the directory would contain:

CDB	LDB	FIELDNAME	TYPE	LENGTH	DESCRIPTION	PDB	etc.
WELLS	LDB1	WELLNUM	TEXT	13	API Well Number	XXX	...
WELLS	LDB1	DEPTH	INT	1	Depth of sample	XXX	...
WELLS	LDB1	POROS	REAL	1	Porosity of sample	XXX	...
WELLS	LDB1	FORMAT	TEXT	VAR	Formation sample found in	XXX	...
WELLS	LDB2	WELLNUM	TEXT	13	API Well Number	YYY	...
WELLS	LDB2	DEPTH	INT	1	Depth of sample	YYY	...
WELLS	LDB2	PERM	REAL	1	Permeability of sample	YYY	...
WELLS	LDB2	WELLNAME	TEXT	VAR	Name of this well	YYY	...

RIM database loading and editing facilities make ADDS database administration a relatively simple task. New CDBs can be added rapidly and made available to the users simply by adding rows to directory relations. As future demands are placed on ADDS, directory extensions will be easy to implement since the entire dictionary/directory is an extendable set of RIM relations.

III. User Interface and User Profiles

The primary functions of the ADDS user interface include:

A. Control all terminal-dependent I/O
B. Provide a human interface that is tailored to the user's needs
C. Control the CDB open process
D. Interact with the user to obtain CDB queries
E. Parse each query into internal representation
F. Check its validity against the directory
G. Filter it into the corresponding "subrequests" to each LDB
H. Represent the query as a set of tasks to be executed in parallel by the DBMS-dependent subrequest servers
I. Pass the query to the ADDS optimizer

Basic ADDS Command Language Translation

ADDS commands are based on an underlying relational query language. The following (Figure 3) is an example of the user interface processing a simple query. The query is issued against the WELLS example CDB (Figure 2) that was introduced in the previous section.

Figure 3. Example "WELLS" Composite Database Query

The user enters the following query:

```
SELECT      WELLNUM WELLNAME DEPTH POROS PERM
FROM        WELLS
MATCHING    WELLNUM
SORTED BY   WELLNUM
WHERE       DEPTH BETWEEN 25000 27000
```

The user interface filters the query and generates one subrequest to LDB1 and one subrequest to LDB2 in the ADDS universal query language:

```
SELECT      WELLNUM DEPTH POROS
FROM        LDB1
SORTED BY   WELLNUM
WHERE       DEPTH GE 25000 AND DEPTH LE 27000

SELECT      WELLNUM WELLNAME DEPTH PERM
FROM        LDB2
SORTED BY   WELLNUM
WHERE       DEPTH GE 25000 AND DEPTH LE 27000
```

The servers are responsible for translating the output of their respective DBMSs into a relational view; therefore, we designed the universal query language around a relational data model.

Full-Screen Query Menus

In the Amoco Production Company Research Center environment, each user is familiar with full-screen computer terminal usage. With this in mind, the ADDS user interface also supports full-screen, fill-in-the-blanks menus with pre-programmed function keys. The full-screen query menus have been tailored to work effectively with the RIM command language. Query menu users only need to know how to fill in the blanks and which program function keys to push. Simple procedures provide individual tailoring of the ADDS environment.

When the system is initiated, ADDS prompts the user for an access password and then checks to see if this user has specified any ADDS-environment "profile" options in a previous session. If a user's profile has defined a default CDB, it is automatically opened. If not, ADDS presents the user with a menu which lists the CDBs that the specified password is authorized to access. The description and purpose of each CDB are displayed, and the user may ask for more detail about the CDB before selecting one to open. When users open a CDB that they have never accessed before, they are presented with a fill-in-the-blanks query menu (Figure 4).

Figure 4. ADDS Full-Screen Query Menu

```
Current CDBNAME - Description of the CDB that is currently open appears here
Query Name-> name        Query Type-> type        Output Relation-> prefix
Description-> The description of this query (if it is cataloged)
Output Fields-> Either "ALL" or a list of fields from CDBNAME
                (optional "=column-width")

Sort Fields-> A list of fields in major to minor sort order
              (optional "=D" for descending sequence)
Match Fields-> A list of JOIN or UNION matching fields

Restrictions-> Any number of WHERE clause row restrictions
               (with Boolean (AND/OR) connectors)




Restriction Operators: EQ, NE, GT, GE, LT, LE, BETWEEN, EXISTS, NULL
Special Names:ROWS,LIMIT  Values:'test',list,integer,real,expression,MAX,MIN
PF1-Invoke This Query     PF2-Get Cataloged Query    PF3-Menu To Change CDB
PF4-EDIT User Profile     PF5-Invoke USER EXEC       PF6-Save Query On FM A5
PF7-Display CDB Fields    PF8-Display CDB Detail     PF9-Help For This Menu
PF10-Next (Older) Query   PF11-Previous Query        PF12-Exit Query Menu
```

For each of the CDBs that a user opens, ADDS "remembers" the most recently entered query. Whenever users re-open a CDB they have queried before, their last query from the previous ADDS session will be redisplayed. The ability to remember the last query of a previous session was designed to assist infrequent users of ADDS in getting back to where they were the last time they used the system. It also saves keyboard time for frequent users.

For application transportability, the ADDS design incorporates a RIM database on each user's private disk space. The default name for these private databases is "ADDS". It is used to store such individualized information as the user's profile, intermediate query results, the parameters of the last query entered by the user for each CDB, and any user-defined "cataloged" queries.

Cataloged queries may also be predefined by the Database Administrator or by user representatives. This type of query is stored, by CDB, in the ADDS dictionary/directory (a RIM database on a common-system disk). Any authorized user may access these CDB-dependent queries.

ADDS facilities to update the RIM databases make it easy for the DBA to catalog new model queries in the ADDS directory or for the users to define their own cataloged queries in their profile database. To catalog a query, the user (or the DBA) starts with a fill-in-the-blanks query menu. The parameters are entered on the query menu, for example, fields to display, row restrictions, and fields to be used for sorting the output rows. The query may be tested by pushing the "invoke" program function key. After displaying the results of the query, the menu is re-displayed with the previous parameters. The parameters may be changed and the new query invoked. When the query produces the desired results, the user may enter an English description of what the query is to be used for and then push the "save" program function key. This action causes the query's parameters to be saved in the user's ADDS database (or the directory).

ADDS users may ask for an alphabetized menu of the names and descriptions of all private and system cataloged queries for each CDB. They may then select and retrieve any query. The current prototype of ADDS also supports scrolling through cataloged queries in chronological sequence, starting with the most recently cataloged query. Once a cataloged query has been displayed, the user may invoke it, modify it, invoke the modified version, and save it in his private ADDS profile database.

With the full-screen menus and a representative set of cataloged queries, database administrators can minimize the education process for new users of each CDB. Users may immediately begin using and tailoring the system to do what they need it to do. RIM to RIM transfers make it easy to pass private cataloged queries to the DBA for inclusion into the directory so that other users may access them. Using RIM as a transportable memory of user personality means that ADDS can assist irregular users to quickly review how they accomplished any previous ADDS operation.

IV. Subrequest Execution

When the optimizer receives a query, it is represented as a set of tasks to be executed in parallel by the DBMS-dependent subrequest servers. The

optimizer must determine if parallel subrequest scheduling is the fastest way to process this request. If it is not, then the optimizer must rearrange the subrequests into the optimum combination of parallel and serialized subrequest scheduling.

The user's view of ADDS response time is highly variable. It obviously depends upon the complexity of the CDB, the speed of the network components and the volume of data that the user has requested. The optimizer can have an important impact on response time. One of the objects of the ongoing ADDS research project is to design and implement several optimization strategies and evaluate their actual performance for a variety of CDBs.

A subrequest scheduling strategy of "maximum parallelism" may occasionally produce the minimum total response time; however, usually, some type of "dependent scheduling" is required. The first ADDS prototype optimizer always scheduled all subrequests in full parallel as a benchmark point.

For advanced optimizers, the directory will need to contain information about the retrieval speed of each data source, transmission speed of each remote link, and summary information about each data source so that the optimizer can estimate the data volume of each subrequest.

The DBMS-specific subrequest servers must be aware of the physical structures of their own data source. They receive one subrequest and translate it into the form that their respective DBMSs understand. They then translate the DBMS output into the two-dimensional relational view that the merger expects from their logical database, and return this output to the monitor. The monitor will store this intermediate server output in the user's private ADDS database where it may be accessed later.

Before the servers can be scheduled, the RIM schema for storing the intermediate results from servers will be dynamically generated. The schema contains one relation for each subrequest, and each relation contains only those fields that are needed. Figure 5 contains the schema required for the example query in Figure 3.

Figure 5. Example Dynamically Generated RIM Schema For Subrequest Storage

SR1 relation in the user's ADDS database:

NAME	TYPE	LENGTH	DEFINITION
WELLNUM	TEXT	13	API Well Number
DEPTH	INT	1	Depth of sample
POROS	REAL	1	Porosity of sample

SR2 relation in the user's ADDS database:

NAME	TYPE	LENGTH	DEFINITION
WELLNUM	TEXT	13	API Well Number
DEPTH	INT	1	Depth of sample
PERM	REAL	1	Permeability of sample
WELLNAME	TEXT	VAR	Name of this well

V. Merger/Formatter

When all subrequests have finished, the monitor invokes the merger/formatter. These routines create the CDB output view by using simple RIM relational algebra operations and/or the BCS/RIM Report Writer. These two important RIM features significantly reduced the implementation time of the first ADDS prototype. In the Figure 3 example, the final output would be generated with a RIM JOIN on the WELLNUM field followed by a RIM SELECT.

The first prototype of ADDS required the users to understand the interactive RIM commands and to use them to perform their own relational algebra and report formatting. For novice users, a future version of ADDS will allow "implied" merging and formatting. This version will require the directory to be expanded to allow the CDB DBA to explicitly specify the relational algebra operations and/or report writer functions that are to generate the final output from the intermediate output of the subrequest servers. The output routines will also have to handle network routing of the formatted output.

Features of the RIM DBMS influenced the choice to use it for server staging and the Merge/Format functions. One difference between RIM and other "implied join" systems, like SQL, is that RIM allows the user to explicitly control the relational algebra operations and to determine how long to keep the intermediate relations. In ADDS, using RIM for intermediate storage is a distinct advantage. Explicit control of relational algebra can reduce re-transmission of data when a series of similar requests are to be entered. The current implementation of the ADDS prototype gives the user this control, but we hope that a future version will be able to remember that it has previously extracted the desired data, thus avoiding re-execution of similar subrequests.

VI. A Transportable Implementation

RIM was selected to speed the initial prototype implementation of ADDS, and to provide a transportable disk I/O manager. We decided to use the RIM Fortran approach to improve transportability of the system code and to simplify the RIM interface. Like RIM, ADDS incorporates a minimum of system-dependent assembler language code, primarily for unsupported functions and where performance is critical. One problem in achieving transportability is a difficulty in exploiting system-dependent hardware, such as multiprogramming environments and unique CRT displays.

The task master, which controls parallel task processors, has a minimal amount of system-dependent assembler code. The subroutines which communicate between parallel tasks (such as scheduler/monitor, task master, and servers) are Fortran-callable assembler language modules.

One of Amoco's sets of enhancements to BCS/RIM exploits the features of the IBM 3270 terminals. To simplify terminal transportability, the ADDS user interface funnels all terminal I/O through a single module, just as RIM funnels all disk I/O through system-dependent modules. A software package that runs on IBM's MVS, VM, and the new PC-XT/370 is used so that 3270-dependent code is at least transportable across all IBM implementations.

The BCS/RIM Fortran interface is incomplete, complex to use, and poorly documented. The availability of the Fortran source code of RIM made it possible to extract subroutines (such as database schema definition) from interactive RIM and to tailor them to the ADDS requirements. Since RIM and ADDS are written in Fortran, the memory requirements are large. This was deemed acceptable in a prototype environment.

As distributed processing continues to flourish, new local area networks of intelligent work stations, file servers, and database machines can use "gateway servers" to connect to one or more local or remote ADDS systems. If a new (terminal or program) view of the data is needed, the database administrator simply defines a new CDB. As new terminal, data storage, or network hardware becomes available, impacted ADDS modules can be modified or new modules added while retaining the original conceptual architecture.

VII. Conclusion

RIM has proved to be useful in speeding the development of ADDS functions such as the dictionary/directory, intermediate subrequest data storage, and the output merger/formatter. The transportable nature of RIM and the availability of its source code have been valuable to the ADDS research project, which is providing an experimental benchmark for a distributed heterogeneous database management system.

APPENDIX

Standard Oil Company (Indiana)/Amoco Production Company

List of Other Types of RIM Applications

I. Literature Catalogs

II. Laboratory Experimental Data

III. Field/Industry Data

IV. Statistical Data Analysis

V. Common Data Interface To a Wide Variety of Existing Programs Including Color Graphical Display/Engineering Systems

VI. Programmer Productivity Tools Including:
Automated Large-Program "DOCPAC" Generation/Collection and Interactive Graphic Display of Subroutine CALL Hierarchy

VII. Satellite-Linked Real-Time Field-Data Collection, Monitoring, and Process Optimization

RIM AS THE DATA BASE MANAGEMENT SYSTEM FOR A MATERIAL PROPERTIES DATA BASE

Patricia H. Karr and David J. Wilson
Martin Marietta Denver Aerospace

SUMMARY

RIM (Relational Information Management) has been selected as the data base management system for a prototype engineering materials data base under development at Martin Marietta Denver Aerospace. The data base will provide a central repository for engineering material properties data, thus facilitating their control. Numerous RIM capabilities are being exploited to satisfy prototype data base requirements. Numerical, text, tabular, and graphical data and references are being stored for five material types. Data retrieval will be accomplished both interactively and through a FORTRAN interface. The experience gained in creating and exercising the prototype will be used in specifying requirements for a production system.

INTRODUCTION

In most engineering organizations, materials information users have typically had to consult a variety of different sources: vendor data, handbooks, laboratory test results, verbal communication, etc. There are several obvious disadvantages to acquiring information in this fashion.

1. The data may vary depending on the source
2. The quality of the data can be suspect
3. Traceability can be a problem
4. The search for information can be time consuming and very expensive
5. Information can be lost through staff turnover
6. Laboratory tests may be conducted or repeated unnecessarily

In response to this state of affairs, a project was begun in 1981 at Martin Marietta Denver Aerospace to provide for centralized control of material properties information through the creation of a computerized engineering materials data base. The work was undertaken as a joint effort by the Materials Section and the Computer-Aided Engineering group of the Engineering Mechanics Department with technical assistance from the Software Engineering Department and from the Data Base Integration organization of Martin Marietta Data Systems. The data base is being implemented on a DEC VAX 11/780 computer under the VMS operating system.

The project has been broken down into several distinct but interrelated phases:

1. Determination of the contents of the data base
2. A preliminary feasibility study
3. Establishment of requirements for a prototype system
4. Selection of the data base manager for the prototype
5. Design and implementation of the prototype
6. Development of final production system requirements based on experience with the prototype
7. Design and implementation of the production system

At present, work is being done on phase five, the prototype phase of the project. This paper will describe how the IPAD Relational Information Management System (RIM-5.0) is being used to address the requirements specified for the prototype system.

NEEDS ASSESSMENT

The first three phases of the project could be viewed as steps in an initial needs assessment process. To determine the contents of the planned data base, questionnaires were circulated in which potential users were asked to specify the materials most critical to their day-to-day work and the most frequently needed information about these materials. The feasibility of implementing a computerized system for storing and retrieving this data was tested by creating a small metals data base using an earlier version of the RIM software (RIM-4.0). This step was helpful in acquainting the project personnel with the capabilities of data base software. It was also valuable in clarifying general system requirements and in making them more realistic. The user surveys and pilot study led to the formulation of the prototype data base requirements. Some of the more important of these were

1. The system must be easy to use
2. It must handle scientific/engineering data types and notation
3. It must support ad hoc queries
4. It should permit several users to make simultaneous queries
5. It should contain a report writer whose use does not require programming experience
6. It must be flexible enough to accommodate future modifications without requiring complete restructuring of the data base
7. It must permit data to be added and updated easily

8. It should have graphics capabilities

9. It should have a reasonable retrieval time

10. It should provide for backup and recovery of data bases

11. It should be able to provide a change log

SELECTION OF THE DATA BASE MANAGEMENT SYSTEM FOR THE PROTOTYPE

At an early stage of the project, the decision was made that a data base management system (DBMS) of the relational type, i.e., one using a two-dimensional-table structure, would probably be preferable to a hierarchical or a network DBMS because of the greater convenience and flexibility inherent in the relational system. The experience gained with the pilot confirmed the wisdom of this decision. After studying available relational DBMSs, it was decided to implement the prototype Engineering Materials Data Base (EMDB) with the latest update of RIM-5.0, even though this version of RIM did not satisfy all of the system requirements. Aside from RIM's relational structure, the following reasons were of greatest importance in this decision:

1. RIM's command language is very simple and straightforward

2. RIM was specifically designed to handle scientific/engineering data types and notation

3. RIM is available on many of the host computers used at Martin Marietta (CDC, VAX, IBM, UNIVAC, and PRIME)

4. RIM allows data bases created on one machine to be readily transferred to another

5. RIM permits one to gain access to data bases either interactively or through an application program

6. RIM is recognized and frequently used by government agencies and by the aerospace industry

It became one of the primary goals of the prototype phase to monitor the evolution of the RIM software in order to assess its viability as the production system DBMS.

PROTOTYPE SCHEMA GENERATION

The current Engineering Materials Data Base is made up of eighteen relations (tables). The primary properties information is contained in five large relations, one for each of the material types in the prototype: 1) adhesives and elastomers, 2) fuels and oxidizers, 3) gases, 4) metals, and 5) plastics. These relations range in size from about 40 to 100 attributes (columns), including a reference attribute for each column of data. A definition relation, i.e., a data dictionary, is provided

for each of the five material types. Data that cannot be included in the properties relations are stored in three supplementary relations called "graphs", "notes", and "tables". Finally, there are five general internal documentation relations containing a change log, a global data dictionary, a list of included materials, a list of references, and descriptions of the data base relations.

Due to the size and complexity of the EMDB, it proved desirable to make use of RIM's powerful sorting and rule checking capabilities in the creation of the schema (structure) for the prototype. The lists of attributes for the five material types included in the system were loaded into a RIM relation so that they could be sorted and examined for consistency of definition and redundancy. The result was a list of all material information attributes in alphabetical order. To this were added the attributes for the above-mentioned internal documentation relations. This master attribute list was then used to generate the definitions of the EMDB relations. Here the RULES feature of RIM proved to be very useful, in that it permitted the establishment and checking of criteria for these relation definitions. One can, for example, assure the proper spelling of the attribute names by requiring that only those attributes preexisting in the master list be allowed to be used in the formation of relations.

The OUTPUT command was used to create a file which, with some editing, was entered into RIM for the actual definition of the data base schema. The preparatory work made it possible to greatly reduce schema generation errors, thus saving much time and labor. It would also be possible to accomplish the creation of the schema through an application program serving as an intermediary between the preliminary relations and the target data base, thereby further automating the process.

LOADING TECHNIQUES

A number of different techniques have been tried for loading data into the EMDB. For the smaller relations, loading is usually done interactively, one row at a time. This permits immediate verification of data entry accuracy, since RIM issues error messages if entered values do not match the attribute definitions as to data type, length, etc. Two methods are used to load large relations. In the first, the data is entered into small "loading relations" which are then combined by means of the RIM INTERSECT command into large "storage relations". Internal array sizes limit the relations so created to about 73 attributes. Relations wider than this have to be loaded by means of INPUT files created and edited outside of RIM. A very helpful feature for the latter method is the ECHO command that causes each input line to be repeated on the CRT screen as it is loaded. If a data entry error occurs, it can be readily found and corrected before a subsequent load attempt is made. It has proved worthwhile to identify "worst-case" examples for each material type which can be pre-loaded into small test relations to check the adequacy of the schema definition prior to full-scale data entry.

MAINTENANCE PROVISIONS

Since RIM-5.0 provides neither a transaction journal nor internal backup and recovery procedures, other means of maintaining traceability of modifications to the data base and of assuring the safety of the data have had to be devised. A simple

transaction journal, which records RIM commands and their results as well as other data about RIM sessions, has been added to the code. To provide for internal documentation of editing activities, a relation was built into the data base schema which provides for recording of old and new values of changed items and other pertinent information. A basic backup and recovery system was accomplished by means of computer command procedures which, when invoked, generate copies of the three RIM data base files and replace damaged data base files with copies of the latest backup files. One of the commercial vendors of RIM has announced its intention of adding a transaction journal capability to its version of the RIM software, which, it is hoped, will eliminate the necessity for the external routines and procedures described above.

EVALUATION OF THE PROTOTYPE DATA BASE

Once the prototype EMDB has been completed, plans call for an evaluation of the system by a variety of users. It is envisioned that these users will make a thorough assessment of the structure, contents, and convenience of the data base and document their reactions to it as well as any problems they encounter while using it. Two methods of making interactive ad hoc queries of the data base will be provided to the evaluators: 1) the RIM command language (primarily the SELECT command with SORTED BY and WHERE clause options), and 2) a set of computer command procedures which allows one to retrieve all of the properties of a given material by responding to prompts displayed on the terminal. Users will also be encouraged to exercise the RIM application program interface to gain access to the EMDB. The user evaluations will be an important element in the development of production system requirements.

CONCLUSIONS AND PROSPECTS

As currently implemented, the prototype Engineering Materials Data Base using RIM-5.0 satisfies many of the requirements originally specified for the system. With the appearance in mid-1982 of commercially supported versions of the RIM software, the remainder of the desired capabilities have been, or soon will be, supplied. Features such as simultaneous read access, plotting, text attribute editing, and a sophisticated report writer are now available. As mentioned above, a transaction journal has been promised for the near future. The new "computed attribute" features, which permit columns of values to be generated through formulas, are sure to be of great utility in storing and retrieving data which vary according to well-defined functions of temperature, pressure, etc., e.g., data on cryogenic materials.

Up to now RIM has, in general, proved to be a powerful, flexible, and easy-to-use data base management system. In this day and age, however, neither user requirements nor data base technology can be expected to remain unchanged for long. A general reassessment of the entire system, taking into account not only the user evaluations discussed above but also recent advances in DBMS capabilities, is called for before deciding on the configuration of the production version of the Engineering Materials Data Base. Regardless of the outcome of this decision-making process, working with RIM has enabled the project participants to gain an appreciation of the power of relational data base management systems and of the role they are destined to play in future engineering data management activities.

N84
22313

UNCLAS

D14

N84 22313

A WIND TUNNEL DATABASE USING RIM

William O. Wray, Jr.
General Dynamics Fort Worth Division

SUMMARY

Engineering database development has become increasingly widespread to industry in recent years with the availability of data management systems. At General Dynamics, a large database has been developed for wind tunnel data and related model-test information, using RIM as the database manager. This paper discusses how the wind tunnel data are arranged into the proper schema for the most efficient database utilization by our engineering user groups. The FORTRAN Interface program of RIM is used extensively in the loading phases of the database and by the users. Finally, several examples are presented to illustrate how the Wind Tunnel Database might be searched for specific data items and test information using RIM.

INTRODUCTION

Wind tunnel investigations generate large amounts of costly data to support many of General Dynamics' research and development projects. Following each investigation, the resulting data are analyzed, reported, and then relegated to some type of data storage. Table 1 illustrates how much data is generated and stored over a period of time.

Because of the large amounts of data, it was determined that a well designed and managed database was needed to maximize the potential usefulness of the wind tunnel data for in-house studies and new business. An early version of RIM, a database management system developed at the Boeing Company under a NASA contract for IPAD, was selected in 1981 for this project because it offered several good features. The most important of these features are listed in Table 2. The continuing support and expected enhancements were a big factor in our selection of RIM. Our primary goal, then, was to establish the database with as much data as was available while attempting to provide the user groups with specialized data in an efficient and precise manner.

SYMBOLS AND ABBREVIATIONS

ALPHA	angle-of-attack
BETA	angle-of-sideslip
CL	lift coefficient
CD	drag coefficient
CM	pitching moment coefficient
CONFIG	configuration number
DEFL-1,2	control surface deflections
MACH	Mach number
QO	dynamic pressure
RIM	Relational Information Management System
RUNNUMb	run number
IPAD	Integrated Program for Aerospace Design

DATA DEVELOPMENT

While developing the schema, it became apparent that a single RIM database would be prohibitively large and not efficient or appropriate for our system. The conclusion, therefore, was that the data should be divided into a number of specialized databases (or files) to reduce both the number of attributes and the number of data rows in each relation. For example; aerodynamic force data consists of three forces and three moments, angles, and a small number of auxiliary data items, while aerodynamic pressure data has no forces but may have 400 pressure data items plus auxiliary data items in each test point.

Fortunately, wind tunnel data has natural divisions which match the disciplines of the database users. This fact allows us to then build many database files which are greatly reduced in size compared to a single database containing all wind tunnel data types and their specific attributes. The natural data divisions and associated model or test information are shown in Table 3. These data types are most common to wind tunnel test work.

The uniqueness of the data types permits designing a RIM datafile tailored for each user group and is generally smaller, highly specialized, and leads to faster response during data searches. An example of the schema for one of the data types is shown in Table 4. This particular schema is for force data and includes four relations. In this database file, the number of attributes per relation does not exceed eighteen, which for this data type is sufficient. Included in the relation 'DATACONS' are the constants used to non-dimensionalize the forces and moments in each test and definitions of up to seven control surface deflections. The actual surface deflections are then located in the relation 'TESTCOND' along with the as-run test conditions for each test. Keys were established only for the 'TESTNUMB' attribute in each relation to speed up the data search operations. Note also that the test data have been divided into two relations by Mach ranges: SUBSDATA (low subsonic) and TRANDATA (high speed; Mach .40 and above). The aerodynamic force and moment coefficient data items appear in these two relations. At some later time, it may become necessary to divide 'TRANDATA' again if the number of data rows become so large that data searching is inefficient for the user. Three additional database files have been created from this database by dividing the data into projects.

A schema similar to the force data was developed for the other data types. The pressure data, inlet data, and store separation data types are such that hundreds of attributes (data items) are generated during these tests. Where possible, we have tried to divide the attributes into relations of less than fifty data items. For pressure data, this was done by creating relations for wing data, fuselage data, vertical tail data, horizontal tail data, etc. Again, the data divisions were made with the end user in mind.

The FORTRAN Interface feature of RIM has been used extensively to load the test data portions of the database. Existing data tapes and mainframe data files from previous wind tunnel investigations were located which showed a number of different data formats amongst the data sources. A series of specialized FORTRAN programs was then written for each data type. These FORTRAN programs featured special routines for different test facilities combined with RIM library subroutines for batch process loading of the data into the appropriate RIM datafile. The loading process has now become somewhat easier because of this program library and the RIM FORTRAN Interface. A small amount of the data and test information must still be entered interactively.

DATABASE USE

Figure 1 shows the database management from data input, through the computer, to the end user. The database users that are familiar with wind tunnel data must only learn the RIM commands to become proficient at interactive searching for data items or performing other database functions. A description of the RIM databases and relations available is shown in Table 5. From this table the user determines where his data may be found (i.e.: Which Database). The 'STATUS'

attribute determines if that database may be immediately accessed or must be restored from tape to the computer first. Extremely large or infrequently used datafiles on the mainframe computer may be in the 'ARCHIVED' category because active storage is very costly for these datafiles over extended periods of time.

After the user is aware of the database contents, the search continues. The Test Summary information in Table 6 helps the user determine which test was conducted for a project, version, or test type of his interest. After locating the appropriate test number, the user next displays the test conditions for that test. These are shown in Table 7 (a typical test-run-log). After locating a specific test condition and run number in this table, the user may then select the desired aerodynamic data items for that run. This report is shown in Table 8. This selected data is now ready for the user to plot, copy, or catalog as a data file for later use, perhaps to incorporate into his FORTRAN programs.

The RIM FORTRAN Interface is also an important feature for the database users who use it for merging these RIM data files into data plot programs and engineering application programs. With the consistent RIM formats, the user need no longer rewrite his application program to be compatible with each new data set.

CONCLUDING REMARKS

Our previous system of preserving wind tunnel data was loosely organized, bulky, redundant, and very slow to search. RIM has permitted us to develop an organized, efficient Wind Tunnel Database for engineering that eliminates most of the previous problems. However, new problem areas have been created which are currently being studied. These include:

1. Storage costs. Who pays for these large database files: the end user or the originating group? An active-inactive file storage system is presently used but does present delays when the inactive data is needed now.
2. Database size. The databases are constantly expanding and may require us to modify schema in the future. The RIM relational algebra commands such as 'JOIN' and 'PROJECT' will enable us to do this easily with a minimum of user associated problems.
3. User education. Engineering user groups must be taught RIM commands and use of the FORTRAN Interface library to make the necessary changes to existing applications programs. A resistance by the users to rewriting existing programs is slowly changing as more and more data become available only in RIM database files.

The announced enhancements to RIM, including plot capability, mathematical manipulation, greater report generating commands, and others, are expected to expand the already useful capabilities of the Wind Tunnel Database for our engineering applications in the years to come.

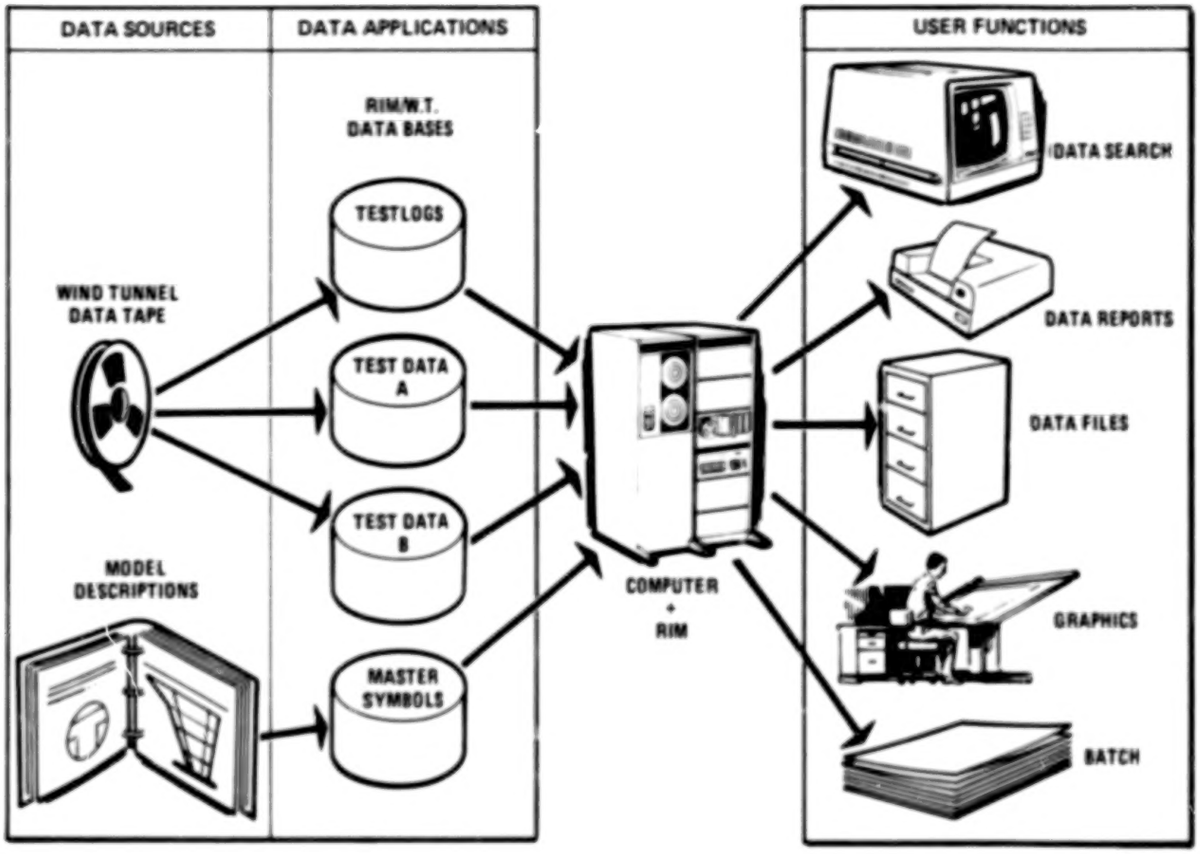


Figure 1.- Wind tunnel data management with RIM.

TABLE 1 WIND TUNNEL TESTS GENERATE MUCH DATA

- WIND TUNNEL TESTS PER YEAR NUMBER 30 TO 35
- DATA ITEMS GENERATED PER TEST: 50,000 OR MORE
- THIS IS OVER 1.5 MILLION DATA ITEMS PER YEAR
- APPROXIMATELY 10 YEARS OF DATA MAY BE ACCUMULATED BY MAJOR PROJECTS
- NEW PROJECTS DERIVE INFORMATION FROM MANY EARLIER PROJECTS
- MAJOR PROJECTS MAY ACCUMULATE 15-18 MILLION DATA ITEMS TO STORE

TABLE 2 IMPORTANT FEATURES OF R/M

1. COMPATIBLE WITH OUR MAINFRAME AND MINI-COMPUTERS
2. ESSENTIALLY UNLIMITED DATA STORAGE
3. DATA BASE SCHEMA EASILY MODIFIED
4. USER FRIENDLY COMMANDS
5. A FORTRAN INTERFACE TO PROVIDE FOR DATA TRANSFER WITH ENGINEERING ANALYSIS PROGRAMS
6. PLANNED FUTURE ENHANCEMENTS

TABLE 3 COMMON WIND TUNNEL DATA TYPES

1. TEST INFORMATION AND TESTLOGS
2. AERODYNAMIC FORCE AND MOMENT DATA
3. MODEL SURFACE PRESSURE DATA
4. STORE LOADS AND CONTROL SURFACE LOADS DATA
5. STORE SEPARATION, DROP, TRAJECTORY DATA
6. INLET AND NOZZLE DATA
7. PROPULSION DATA
8. MODEL DIMENSIONS AND DESCRIPTIONS

TABLE 4 RIM SCHEMA FOR A TYPICAL DATA TYPE

DATA BASE NAME: DFORCE1					
RELATION NAME	RPW	MPW	MODIFY DATE	NO. ATTRIBUTES	NO. ROWS
DATACONS	NONE	YES	09/20/83	16	24
TESTCOND	NONE	YES	09/20/83	18	4707
SUBSDATA	NONE	YES	09/20/83	15	21906
TRANDATA	NONE	YES	07/29/83	15	82393

RELATION : TESTCOND		RELATION : DATACONS		RELATION : SUBSDATA		RELATION : TRANDATA	
NAME	TYPE	NAME	TYPE	NAME	TYPE	NAME	TYPE
PROJECT	TEXT	TESTNUMB	TEXT	TESTNUMB	TEXT	TESTNUMB	TEXT
FACILITY	TEXT	FIRSTRUN	INT	RUNNUMB	INT	RUNNUMB	INT
TESTNUMB	TEXT	LASTRUN	INT	CONFIG	INT	CONFIG	INT
RUNNUMB	INT	SREF	REAL	POINT	INT	POINT	INT
MACH	REAL	CBAR	REAL	MACH	REAL	MACH	REAL
Q0	REAL	SPAN	REAL	ALPHA	REAL	ALPHA	REAL
RN[illegible]T	REAL	CG-%CBAR	REAL	BETA	REAL	BETA	REAL
[illegible]/D	REAL	INC-WING	REAL	CL	REAL	CL	REAL
CONFIG	INT	DATAMULT	REAL	CM	REAL	CM	REAL
RUNTYPE	TEXT	DEF-1	TEXT	CD	REAL	CD	REAL
TESTDATE	INT	DEF-2	TEXT	CY	REAL	CY	REAL
DEFL-1	REAL	DEF-3	TEXT	CLNS	REAL	CLNS	REAL
DEFL-2	REAL	DEF-4	TEXT	CLLS	REAL	CLLS	REAL
DEFL-3	REAL	DEF-5	TEXT	CWBM	REAL	CWBM	REAL
DEFL-4	REAL	DEF-6	TEXT	CACC	REAL	CACC	REAL
DEFL-5	REAL	DEF-7	TEXT				
DEFL-6	REAL						
DEFL-7	REAL						

TABLE 5 DESCRIPTION OF DATA BASES AND RELATIONS

DATABASE	RELATION	NUM/ATTR	STATUS	DESCRIPT
TESLOG	TESTS	17	ACTIVE	PROJECT, TESTDATE, FACILITY, MACHS, TESTHOURS, DATATYPE, KEYWORDS
TESLOG	CONFVARI	23	ACTIVE	MODEL SYMBOLS TESTED, TESTNUMBER, DEFLECTION DESCRIPTION
TESLOG	TESTLOG	23	ACTIVE	TEST RUNLOG LISTING, CONDITIONS, DEFLECTIONS, REMARKS, TESTNUMB
TESLOG	SHORTEST	43	ACTIVE	RUNLOG FOR STORETESTS, CONDITIONS, STORES, PYLONS, LOCATIONS
DFORCE1	TESTCOND	18	ACTIVE	RUNLOG FOR CONDITIONS, DEFLECTIONS, F-16A/B/C/FORCE TESTS
DFORCE1	DATACONS	16	ACTIVE	REFERENCE CONSTANTS, DEFLECTION DESCRIPTIONS
DFORCE1	SUBSDATA	15	ACTIVE	F-16A/B/C FORCE DATA, ALPHA, BETA, MACH, TEST, RUN, PT (LOWSPEED)
DFORCE1	TRANDATA	15	ACTIVE	F-16A/B/C FORCE DATA, ALPHA, BETA, MACH, TEST, RUN, PT (HI-SPEED)
DFORCE1	INTLDATA	10	ACTIVE	F-16A/B/C DUCT INTERNAL DRAG DATA, AO/AI, PRESS RECOVERY
DFORCE2	TESTCOND	18	ACTIVE	RUNLOG FOR CONDITIONS, DEFLECTIONS, F-16XL FORCE TESTS
DFORCE2	SUBSDATA	15	ACTIVE	F-16XL FORCE DATA, ALPHA, BETA, MACH, TEST, RUN, PT (LOWSPEED)
DFORCE2	TRANDATA	15	ACTIVE	F-16XL FORCE DATA, ALPHA, BETA, MACH, TEST, RUN, PT (HI-SPEED)
DFORCE2	DATACONS	16	ACTIVE	F-16XL TEST REFERENCE CONSTANTS, DEFL. DESCRIPTIONS
DFORCE3	TESTCOND	18	ARCHIVED	RUNLOG FOR CONDITIONS, DEFLECTIONS, AFTI-16 FORCE TEST
DFORCE3	DATACONS	18	ARCHIVED	AFTI-16 TEST REFERENCE CONSTANTS, DEFLECTIONS DESCRIPTIONS
DFORCE3	SUBSDATA	18	ARCHIVED	AFTI-16 FORCE DATA ALPHA BETA MACH TEST RUN, PT (LOWSPEED)
DFORCE3	TRANDATA	18	ARCHIVED	AFTI-16 FORCE DATA ALPHA BETA MACH TEST RUN, PT (HI-SPEED)
DFORCE4	TESTCOND	18	ARCHIVED	RUNLOG FOR CONDITIONS, DEFLECTIONS
DFORCE4	TESTCOND	18	ARCHIVED	RUNLOG FOR CONDITIONS, DEFLECTIONS – ALL NON F-16 TESTS
DFORCE4	DATACONS	16	ARCHIVED	MISC AIRCRAFT TEST REF CONSTANTS, DEFLECTION DESCRIPTIONS
DFORCE4	SUBSDATA	15	ARCHIVED	MISC AIRCRAFT FORCE DATA ALPHA, BETA, MACH, TEST, PT (LOWSPEED)
DFORCE4	TRANDATA	15	ARCHIVED	MISC AIRCRAFT FORCE DATA ALPHA, BETA, MACH, TEST, PT (HI-SPEED)
DLOADS	TESTCOND	18	ARCHIVED	ALL LOADS TESTS RUNLOG FOR CONDITIONS, DEFLECTIONS
DLOADS	F16ADATA	17	ARCHIVED	F-16A/B/C LOADS DATA, MACH, ALPHA, BETA, RUN, BALANCE (LOC)
DLOADS	F16EDATA	17	ARCHIVED	F-16E/XL LOADS DATA, MACH, ALPHA, BETA, RUN, BALANCE (LOC)
DLOADS	DATACONS	16	ARCHIVED	ALL LOADS TESTS REFERENCE CONSTANTS, DEFLECTION DESCRIPTION
MSYMBL	PROJ-SYM	27	ACTIVE	ALL PROJECT MASTER SYMBOL LETTER DEFINITIONS
MSYMBL	GENERAL	35	ACTIVE	MASTER SYMBOL DATA REFERENCES, DIMENSIONAL CONSTANTS, PROJECTS
MSYMBL	STOREDAT	14	ACTIVE	MASTER SYMBOL STORE DATA, DIMENSIONS, SHAPES, LOCATION, NAME
MSYMBL	AIRFOILS	14	ACTIVE	MASTER SYMBOL AIRFOIL DATA, STATIONS, DIMENSIONAL INFO

TABLE 6 A SUMMARY OF TESTS

PROJECT	TESTTYPE	VERSION	FACILITY	TESTNUMB	DATABASE	TESTRANG	TESTHRS	REPORT
F-16	FLUTTER	A/B	LARC16TD	LD160345	N/A	TRAN	128.	NONE
F-16	FLUTTER	E	LARC16TD	LD160344	N/A	TRAN	32.	NONE
F-16	FLUTTER	E	LARC16TD	LD160343	N/A	TRAN	72.	NONE
RESEARCH	PROPUL	POWERED	NSRDC	NS073361	PROPUL	TRAN	48.3	FZT446
RESEARCH	PRESS	POWERED	NSRDC	NS073361	DPRESS	TRAN	48.	FZT446
F-16	FORCE	AFTI	GDLST	LS127664	DFORCE	LOW	12.	20PR124
F-16	LOADS	AFTI	GDLST	LS127664	DLOADS	LOW	4.5	20PR124
F-16	FORCE	A/B	GDLST	LS128030	DFORCE	LOW	1.8	FZT442
F-16	FORCE	A/B	GDLST	LS128020	DFORCE	LOW	21.	FZT442
F-16	FORCE	AFTI	GDLST	LS127663	DFORCE	LOW	16.6	20PR124
F-16	LOADS	AFTI	GDLST	LS127663	DLOADS	LOW	10.	20PR124
F-16	FORCE	A/B	GDLST	LS127980	DFORCE	LOW	50.9	16PR2092
F-16	LOADS	E	CAL-8	CL803663	DLOADS	TRAN	5.	400PR033
F-16	FORCE	E	CAL-8	CL803663	DFORCE	TRAN	18.5	400PR033
F-16	FORCE	E	CAL-8	CL803653	DFORCE	TRAN	52.5	400PR031
F-16	LOADS	E	CAL-8	CL803653	DLOADS	TRAN	12.5	400PR031
F-16	FORCE	A/B	PWT-16T	TF160604	DFORCE	TRAN	22.5	16PR2126
F-16	FORCE	A/B	PWT-16T	TF160604	DFORCE	TRAN	3.6	FZT442
F-16	FORCE	A/B	PWT-16T	TF160604	DFORCE	TRAN	14.9	FZT442
F-16	FORCE	A/B	PWT-16T	TF160604	DFORCE	TRAN	42.3	16PR2099
F-16	FORCE	AFTI	PWT-16T	TF160604	DFORCE	TRAN	2.7	20PR124
F-16	LOADS	AFTI	PWT-16T	TF160604	DLOADS	TRAN	2.7	20PR124
F-16	FORCE	F	GDLST	LS127845	DFORCE	LOW	73.2	400PR028
F-16	FORCE	E	LARC-4	LA041380	DFORCE	SUPER	48.	400PR029
F-16	LOADS	E	LARC-4	LA042380	DLOADS	SUPER	22.	400PR029

TABLE 7 A TYPICAL TEST-RUN-LOG

PROJECT	FACILITY	TESTNUMB	RUNNUMB	MACH	QO	RUNTYPE	DEFL-1	DEFL-2
F-16	GDLST	LS128020	7	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	1	.18	50.	PITCH	-2.	0.
F-16	GDLST	LS128020	2	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	3	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	4	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	5	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	6	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	15	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	16	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	17	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	18	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	19	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	20	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	21	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	22	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	23	.2	60.	YAW	0.	0.
F-16	GDLST	LS128020	8	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	9	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	10	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	11	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	12	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	13	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	14	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	31	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	32	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	33	.2	60.	PITCH	30.	0.
F-16	GDLST	LS128020	34	.2	60.	PITCH	20.	0.
F-16	GDLST	LS128020	35	.2	60.	PITCH	10.	0.
F-16	GDLST	LS128020	28	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	29	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	30	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	27	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	24	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	25	.2	60.	PITCH	0.	0.
F-16	GDLST	LS128020	26	.2	60.	PITCH	0.	0.

TABLE 8 AERODYNAMIC DATA ITEMS SPECIFIED BY USER

RUNNUMB	CONFIG	POINT	MACH	ALPHA	BETA	CL	CM	CD
35	4	4	.2	.1014	.0008	-.0394	-.0387	.0332
35	4	5	.2	-.9483	.0006	-.109	-.0462	.0389
35	4	6	.2	.1029	.0003	-.0377	-.0384	.0331
35	4	7	.2	1.1745	.0007	.0341	-.0303	.0295
35	4	8	.2	2.2917	.0008	.1129	-.0225	.0271
35	4	9	.2	3.2972	.001	.1776	-.0152	.0287
35	4	10	.2	4.4106	.0013	.2513	-.006	.029
35	4	11	.2	4.4118	.0014	.2523	-.0056	.0315
35	4	12	.2	5.4726	.0014	.5225	.0039	.0366
35	4	13	.2	6.5832	.0014	.3921	.0139	.0433
35	4	14	.2	8.619	.0015	.5238	.0349	.0615
35	4	15	.2	10.8115	.0025	.6624	.0577	.0897
35	4	16	.2	16.0315	.0041	.9765	.125	.211

D15

N84 22314

PRODUCT DEFINITION DATA INTERFACE

Burt Birchfield and Peter Downey
McDonnell Douglas Corporation

INTRODUCTION

The U.S. Aerospace Industry has been a leader in developing and applying advanced Computer Aided Design/Computer Aided Manufacturing (CAD/CAM) technology. New CAD/CAM capabilities are evolving at an increasing rate to provide the engineer and production worker with tools to produce better products and significantly improve productivity. This technology is rapidly expanding in all phases of engineering and manufacturing with large potential for improvements in productivity. However, the national challenge is to integrate CAD and CAM systematically to insure maximum utility throughout the U.S. Aerospace Industry, its large community of supporting suppliers, and the Department of Defense aircraft overhaul and repair facilities. In meeting this challenge, the need is evident for a framework for exchange of digital product definition data, which serves the function of the conventional engineering drawing.

IGES (Initial Graphics Exchange Specification), lead by the National Bureau of Standards and developed in cooperation with key industrial and government organizations, has established the initial base for direct digital exchange of graphics data. IGES 1.0 has been incorporated into the ANSI Y14.26M, Section 2-4 (American National Standards Institute) standard and is currently in the process of implementation by major vendors and users. IGES is the current answer to realization of the potential benefit of direct interface of digital part definition among graphics systems.

However, there is a need to look at the future of manufacturing and the end state needs for "complete product description" which will serve as the input to the factory of the future. Advanced manufacturing technologies in numerical control, robotics, automated process planning, inspection, and the integration of these into a cohesive system are fundamentally limited without the definition of the product in a manner which directly feeds these processes.

PROJECT OBJECTIVE

The objective of the PDDI project is twofold as shown in Figure 1. Task I will establish the current state of ANSI/IGES implementation through the application of test procedures against current graphics systems. The major emphasis of the project is in Task II - a definition of long-range manufacturing needs and the demonstration of a prototype product definition data interface to meet these needs. This system is intended to serve as the information interface between engineering and all manufacturing functions using today's blueprint, including process planning, numerical control (N/C) programming, quality assurance, tool design and others. The Product Definition Data Interface (PDDI) will be demonstrated together with an advanced Numerical Control Programming system and an advanced Process Planning system in order to prove the concept. In addition, it will be interfaced to two commercial CAD systems to demonstrate its general applicability.

Key summary objectives for Task II as the major emphasis of the project are:

- A thorough analysis of manufacturing information needs for Product Definition Data (PDD) using sample aerospace parts.
- Definition of an automated framework for a Product Definition Data Interface (PDDI).
- Development of data format and utilities required to support the PDDI.
- Proof of concept of the PDDI through demonstration of the utility software.

PROGRAM SCHEDULE AND SUMMARY

Tasks I and II were conducted concurrently in contract options and steps as outlined in Figure 2 - Program Schedule, and Figure 3 - Program Summary.

Task I was a 12 month program phased into a basic program and an option. The basic program developed ANSI/IGES test files for sample parts and test procedures for the testing and evaluation of IGES implementation. The major deliverable of this program phase was a System Test Plan. Option 1 involved the actual testing of ANSI/IGES implementation at 12 vendor/user sites using the system test plan and files. The key deliverable from the option was a System Test Report on the general level and usability of IGES within the current CAD graphics systems environment. This report is non-attributive in nature relative to specific systems. It focuses on the general level of implementation and usability of IGES as a communication medium for digital product description. The results of Task I are a key input to Task II activities. Because IGES and its implementation are in a constant state of extension and improvement, the Task I Final Report provides not only Task I results, but also a System Test Plan for continuing use in the IGES program and general industry. The Task I effort has been completed.

Task II is a 33 month program which is phased into a 9 month Basic Program for determination of manufacturing needs for future PDDI, a 12 month Option 1 for assessment of the state-of-the-art and establishment of system needs, and a 12 month Option 2 for design, implementation, and demonstration of a PDDI prototype.

The Task II approach (Reference Figure 3 - Program Summary) follows the ICAM Life Cycle methodology for system development. The program scoping strategy, as outlined by the Air Force, is shown in Figure 4. The initial emphasis is on a broad view of manufacturing needs with focus on a "walk-through" of sample parts through the manufacturing functions of the ICAM MFGO Architecture. This details the uses of Product Definition Data in the current manufacturing environment and provides a basis for the scoping of manufacturing needs for Product Definition Data in the future manufacturing environment. In addition to the parts walk-through, the program is making maximum use of external inputs from industry, ICAM program, NASA IPAD (Integrated Program for Aerospace Vehicle Design), and CAM-I (Computer Aided Manufacturing-International). These needs have been associated with benefits and have been prioritized. A deliverable of the Basic Program was a Needs Analysis document. This served as an input to the State-of-the-Art (SOA) assessment and later program phases. Through a continuing scoping process based on needs versus available technology, the final prototype will demonstrate principles of a future PDDI using the sample parts.

One main objective of the PDDI prototype is to provide both man and computer understandable data for use in advanced manufacturing technologies such as process planning and N/C programming.

PROJECT ORGANIZATION

McDonnell Aircraft Company (MCAIR) has provided an integrated engineering/manufacturing team to execute the PDDI program (Figure 5). A program manager and deputy program manager have been assigned from engineering and manufacturing organizations with a manufacturing program manager, Peter J. Downey, during manufacturing requirements definition phases and an engineering program manager, Edwin B. Birchfield, during implementation phases. This assures maximum usability of the PDDI results in both engineering and manufacturing.

Task I Subcontractor - Booz, Allen, & Hamilton was the principal subcontractor to MCAIR in Task I IGES evaluation efforts. They had the primary role in development of the test procedures and methodologies and were responsible for the administration and analysis in the testing of twelve IGES systems.

Task II Team - Task II work is being centered in MCAIR teams of engineering and manufacturing personnel. In addition to systems personnel, a key strategy of the program is the direct involvement of manufacturing and engineering users responsible for design, production planning, tooling and quality assurance of aerospace parts.

MCAIR/MDC Advisory Boards - In addition, program results are undergoing constant review by MCAIR and McDonnell Douglas Corporation advisory boards consisting of key line and system development managers.

Subcontractor for Turned Parts - United Technologies Research Center (UTRC) is the principal subcontractor to MCAIR for turned part expertise. UTRC is known for its expertise in development of geometric modeling and generative process planning systems for turned parts.

Industry Review Board - An important aspect of the PDDI Program was the establishment of an Industry Review Board (IRB) consisting of ten key aerospace airframe and engine manufacturers and subcontractors, as well as automotive representatives. The IRB is chaired by James F. Lardner, vice president, Advanced Manufacturing Development, of Deere and Company. The IRB meets quarterly to review project progress and results, and provides the input required for general application of the PDDI program results throughout U.S. industry.

Task I Working Group - A special Task I Working Group, consisting of members from IGES committees, was established for close coordination on Task I activities. This working group was chaired by Brad Smith, IGES Program Manager of the National Bureau of Standards. As a result of this close coordination with the IGES community, the system test plan, procedures, and test files of Task I are an important output for continued use within the IGES and general industry communities.

SAMPLE PART WALK-THROUGH APPROACH

A significant strategy of the PDDI Program was the selection and manufacturing walk-through of sample aerospace parts representative of typical aerospace sheet metal, composite, machined, and turned-part classes. These sample parts are being

used both in Task I and Task II activities. They are a focal point for current IGES methodology and future PDDI needs for transfer of all necessary part definition information - including geometry, material, dimensions, tolerances, features, and engineering instructions.

Part Selection - The sample parts selected for the PDDI Program are typical airframe assemblies from the F-18 leading edge extension (LEX) and trailing edge flap (Figure 6). The sheet metal rib, machined rib, and sheet metal skin have been selected from the F-18 LEX (Figures 7 and 8). The parts selected from the trailing edge flap include a composite rib and a composite skin (Figure 9). This selection process provides the basis upon which to compare similarities and differences between sheet metal, machined and composite ribs, and a sheet metal and composite skin class of part. The sample turned part was selected from prior work performed in the CAM-I (Computer Aided Manufacturing-International) geometric modeling project (Figure 10). An objective of the program is to establish generic capability required for all classes of parts as well as class specific requirements for individual part classes.

Use of ICAM MFG0 Architecture - The ICAM architecture MFG0 was used as the basis for identification of all manufacturing functions utilizing today's part drawing and engineering instructions which are released by engineering departments. The existing system was documented, including all engineering information included on today's part drawing, and manufacturing uses of that information. This information includes Bill of Material data, coordinate systems, part and reference geometry, tolerances, notes, and material and process specifications. An extension to the current engineering drawing in the walk-through analysis was the definition of "part features" which are currently interpreted from the engineering drawings. Recognition of part features, such as flanges, webs, joggles, cutouts, bends, pockets, etc., is the key to current manufacturing decision making processes such as process planning, tooling, and N/C programming. Classification-coding approaches have implicity used part features as a basis for classifying part families from which automated decisions can be made. A path to future manufacturing automation is through the direct recognition of part features in the automated Product Definition Data. The ICAM MFG0 architecture clearly identifies early design-manufacturing interaction long before the final "released drawing" is received from engineering. The manufacturing needs analysis included a broad view of manufacturing needs throughout the MFG0 architecture including early development of manufacturing plans, estimating requirements of time and cost to produce, making of schedules and budgets, the providing of production resources, and assembly requirements. However, focus was on the detail needs for part definition, detail part planning, tooling, N/C programming, and advanced technologies. These focus areas have been analyzed in detail using the structured IDEF0 (function modeling) and IDEF1 (information modeling) methodologies. This provided a basis for analysis of current manufacturing methods, systems, and needs for future PDDI definition.

Analysis of Current Drawing Information - The conventional engineering drawing for the sample sheet metal rib is shown in Figure 11a. A detailed analysis of this drawing shows that it contains much more than just the information required to define the part. Information contained on the face of the drawing includes loft and design data reference (Figure 11b) as well as drawing field blocks to indicate configuration of loft information. It contains layout and reference part data (Figure 11c) from mating parts as well as drawing field reference to the layout and mating part drawings. It contains information required for flat pattern definition (Figure 11d) of this part and it contains part forming data required for the formed definition (Figure 11e) of this part. It contains hole locations, partial fastener information,

and hole mate with data (Figure 11f) for mating parts called out on the field of the drawing. This type of drawing is typical of aerospace structure, providing not only the definition of the detailed part, but also the reference information for its overall assembly environment. The current engineering drawing provides a large amount of valuable information. But this information is structured in a man readable format through drawing field notes, which do not lend themselves to automated processing.

The Need for Future PDD - Future manufacturing needs are for complete three-dimensional (3-D) product description and man-computer understandable part definition in terms of part features such as flanges and holes and associated reference data currently contained on the engineering drawing.

Needs for Future PDD - A Conceptual Framework - Although focusing on detailed part definition, it is important to envision an overall strategic framework within which the overall 3-D product could be defined. The following scenario provides a conceptual framework for evolutionary release of engineering data and a conceptual framework for assemblies using the F-18 LEX assembly. Figure 12 shows the overall assembly envelope of the LEX and key attach points. This information provides key input to initial planning for manufacturing and also is early engineering data required for commitment to long lead tooling such as master models as shown in Figure 13. Figure 14 shows an advanced concept for definition of design data for location of major structure. This provides input to manufacturing functions in assuming a structure and method of manufacture of major configurations and line assemblies, and development of support activities plans. This is the information required for commitment to long lead tooling, such as assembly jigs (Figure 15). Figure 16 shows more detailed design data including location of major substructure. Association of key material and potential early part number assignment to this type of data provides input to manufacturing functions such as estimating requirements for time and cost to produce, development of production plans for assemblies and detailed parts, and the initial making of schedules and budgets. Figure 17 shows a more detailed assembly skeletal which may be the engineering layout drawing of the future. With the addition of part features, such as webs and flanges, along with the association of material and early part number identification, manufacturing could begin the development of batch assembly sequencing and the commitment of production resources in facilities, equipment, and tools. Skeletal data of this nature could provide major input to generation of classification-code information which could, in turn, be keyed to producibility, cost estimating, and make/buy information. A skeletal assembly model, (Figure 18), provides the visualization which manufacturing can use for detailed part features, such as flanges, joggles, estimates of number of fasteners, and detailed tools. This skeletal concept provides for not only evolutionary release of data, but also the potential for long term replacement of the "assembly drawing" in an automated environment. As manufacturing technologies move more toward graphics CRT based systems, it is a critical human factor that a man can retrieve and analyze only the level of detail that he requires without being overwhelmed with detail and the computer response required to generate detailed displays. Communication and control of part joints provides a basis for fit analysis and control. Early availability of reference data, such as shown in this figure, could be directly scribed onto N/C master models which may be committed before full part definition is available. This also provides a conceptual framework for robotics drilling and trimming and storage of information such as joint thickness, related hole depths and fasteners, and fastener grip length information.

Needs for Part Definition - A Detailed Analysis - The focus of project activities have been on the data directly related to the detailed analysis of the detail sample parts via the ICAM IDEF0 and IDEF1 methodologies. Figure 19 outlines the general methodology used including: 1) identification of key MFG0 functions (IDEF0 Diagrams) related to detail part planning, tooling, fabrication, and inspection; 2) collection of all current production documents as source material; 3) identification of Product Definition Data elements contained on the documents and used in the functions; 4) categorization of entities into general classes; 5) development of relationships among classes; 6) development of IDEF1 diagrams of entities, attributes and relationships; 7) identification of cost/performance drivers relative to use of data in serving manufacturing functions.

This served as the basis for the Needs Analysis Document and summary information matrices comparing generic and part class specific data needs for the sample parts.

Figure 20 shows the concept of carrying assembly fit interfaces to more detailed skeletal "envelope surfaces" of an individual part, including information such as mating surfaces and joggles. This "envelope" of data provides simple visualization of key part features, without need for display of complete part detail. Figure 21 shows the concept of inclusion of the inside surfaces on the sheet metal rib. Availability of this data provides for input to form block design, and N/C programming of form blocks as well as providing simple thickness visualization of the part. Figure 22 shows a fully detailed sheet metal part including full definition of trim, holes, and bend radii. This provides data required for more detailed analysis of holes and trim, preparation of visual aids for assembly, robotics drill and trim applications, and elimination of interpretation problems.

Although PDDI is focusing on a replacement for the function of the engineering drawing, the project is also examining provisions for growth paths for use of PDDI for representation and associativity of sub-part forms generated in the manufacturing/planning process. This particular sheet metal rib was planned for fabrication "less holes" in the flanges with these holes to be drilled upon final assembly. Association of these sub-part forms to a copy of the original engineering part description (Figure 23) would provide a growth path for use for PDDI as a communication device among manufacturing functions. Figure 24 shows a potential use for PDDI data such as required for automated nesting and automated form block design using key manufacturing subpart shapes.

Summary of PDDI - Replaces the Function of an Engineering Drawing - Figure 25 shows the overall framework for potential use of PDDI as a communication vehicle for replacement of the function of an engineering drawing. The current product definition data system for a total product within a company typicaly consists of an engineering Bill of Material data base which is linked to either an automated or drawing based geometric data base. The Bill of Material is typically linked to a manufacturing Bill of Material via a direct computer linkage. It is beyond the scope of PDDI to act as an overall data base management system. But PDDI is targeted for replacement of the function of an engineering drawing as a means of communicating engineering data including drawing/part number identification, parts list data, part geometry definition, accommodation of reference geometry, material, drawing notes, and identification of process specifications. The general scoping strategy is to use the current engineering drawing standard, within aerospace, DOD-STD-100C, as the basis for identification and control of product definition data. The choice of this current standard provides maximum compatibility with manufacturing systems, customer systems, and vendor systems in communication of the automated drawing replacement.

Approach to Needs Analysis/Prioritization - In approaching the benefits and prioritization process, the given PDDI objective was to complete part definition in the providing of all information for all manufacturing functions in an automated format. The two key goals for the benefits analysis provide for (1) man understanding of PDD and, (2) a computer understanding of PDD as input to automated manufacturing functions such as process planning and N/C. However, the final PDD goal of complete part definition will by its nature be an evolutionary process from current wireframe technology through surfaces, to association of functional data, to potential constructive geometry techniques, to complete solid part definition. Needs were categorized into these information levels and prioritized. We anticipate an evolutionary framework for aerospace which addresses complex surfaces, constant thickness dimensions and tolerances related to those surfaces, surface connectivity methods, "faces" which allow for complex bounding and representation of holes in surfaces, connectivity of faces, structuring of faces into part features and regions such as flanges, webs, pockets, holes, etc., and finally, a complete part definition which captures all information in both a man and computer understandable format.

Option 1 - State-of-the-Art Survey - The Needs Analysis and Task I IGES test results were used as inputs to the state-of-the-art assessment. Visits were made to major aerospace companies and graphics systems vendors, and reviews were made of current IGES standard and research programs related to Product Data Definition. The State-of-the-Art Survey established the current and projected state-of-the-art to be used as input to later project phases for system requirements and prototype development. Public domain developments such as those in ICAM, IPAD, IGES, and CAM-I programs were especially considered for maximum applicability to industry.

Software Deliverables - The target project deliverables and their possible demonstration use are outlined in Figure 26. The primary deliverables will be:

- A PDDI Exchange Standard Format for communication of Product Definition Data.
- A Working Form of the PDDI format to be used in the prototype demonstration.
- Access Software to be used in the demonstration. This access software is targeted to be simple in form, but to provide assistance to external systems in development of simplified interfaces to extract the appropriate types of information required by these systems.

Final Report - The final report of the project is targeted for final submittal to the Air Force in December of 1985.

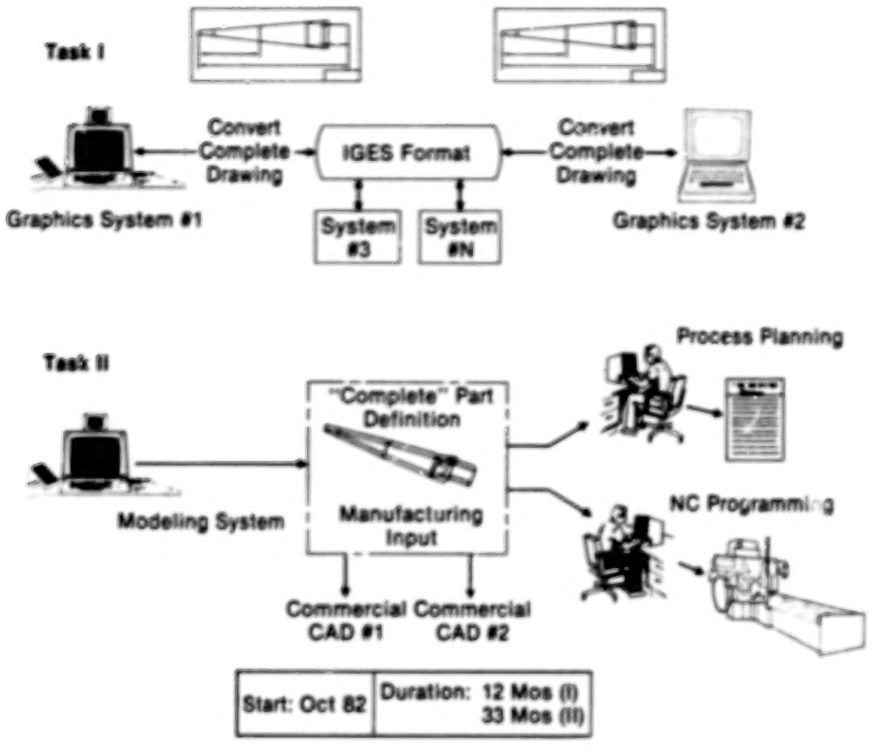


Figure 1. - ICAM product definition data interface (PDDI).

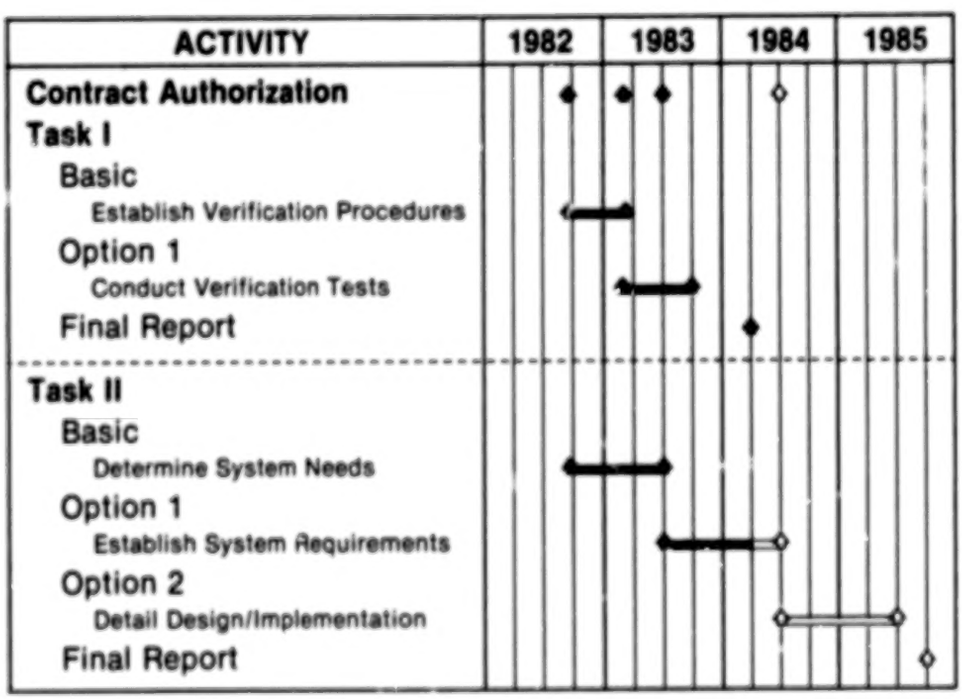


Figure 2. - PDDI schedule - tasks I and II.

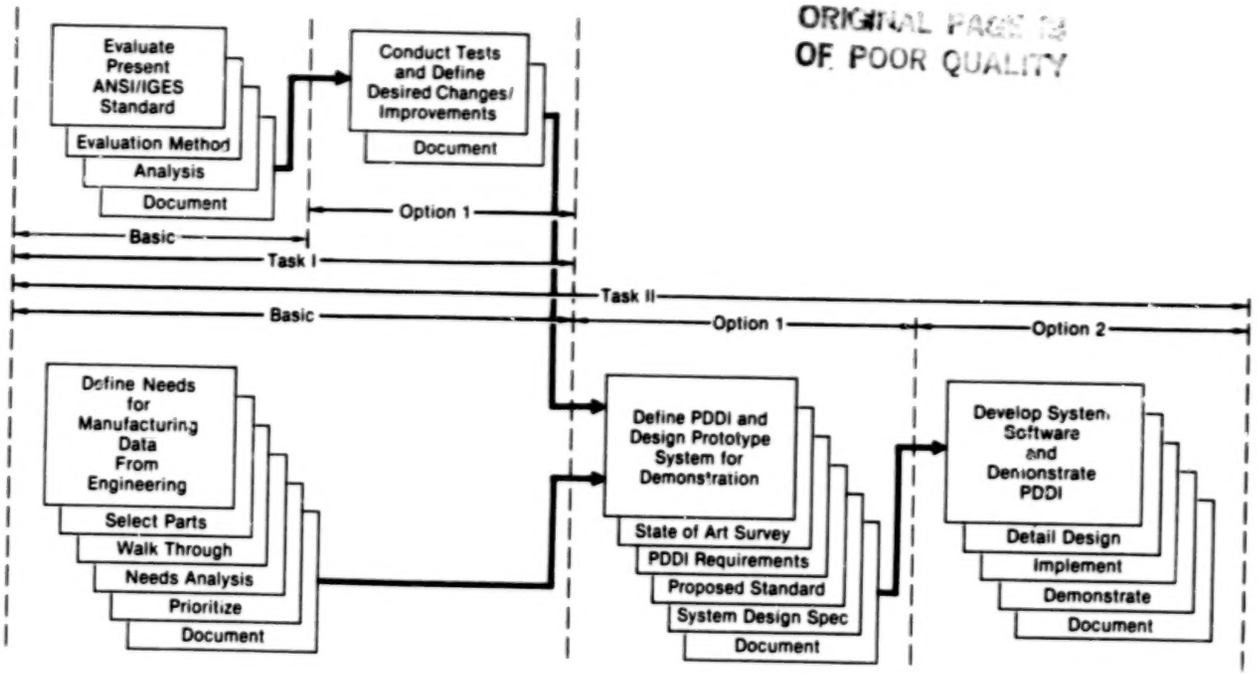


Figure 3. - Program summary.

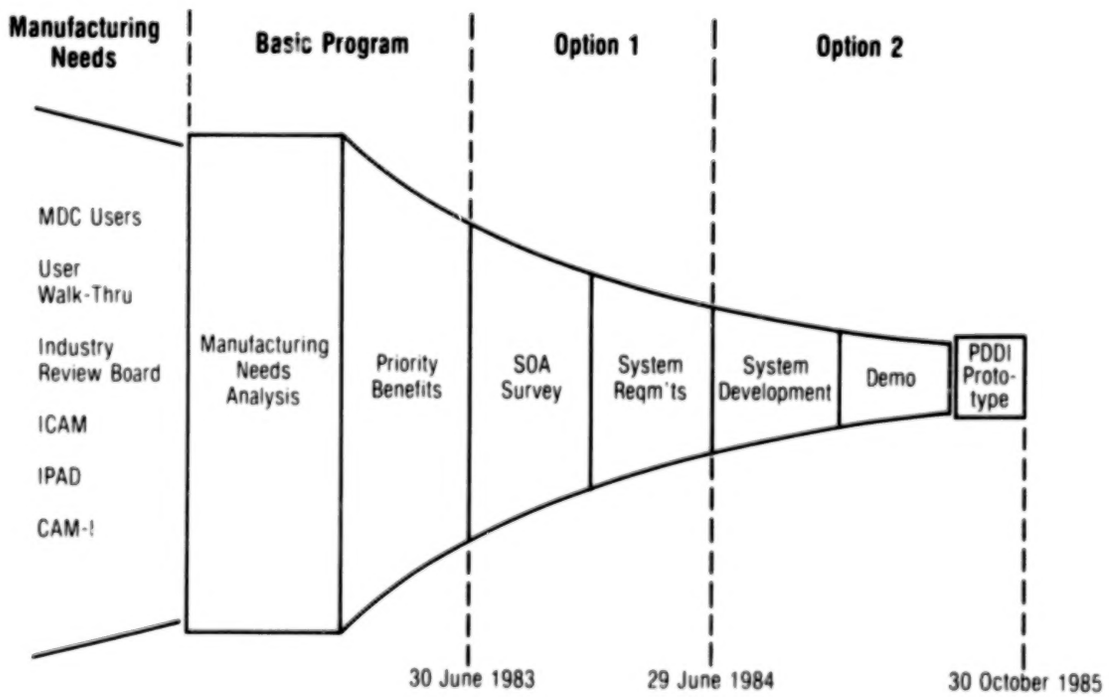


Figure 4. - Air Force PDDI scoping approach.

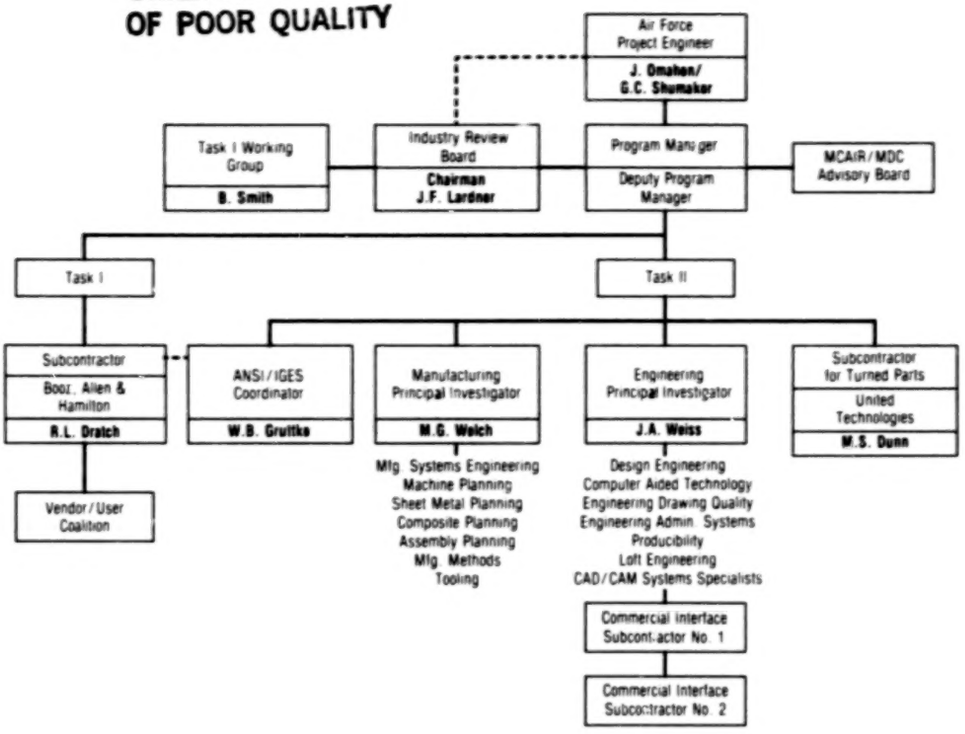


Figure 5. - PDDI project organization.

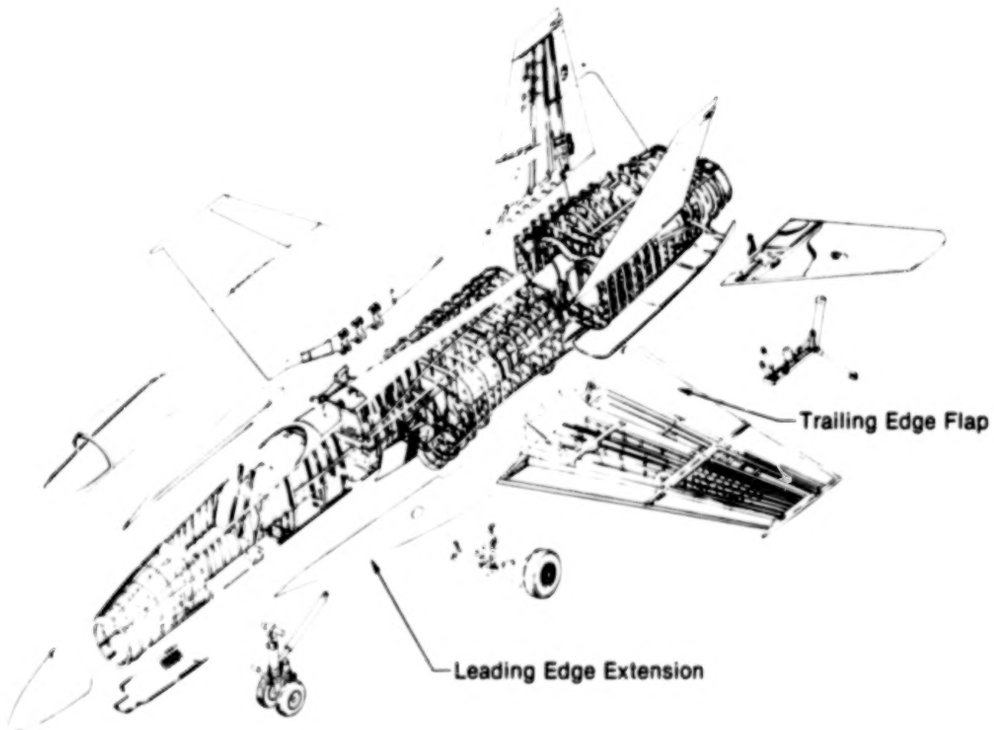


Figure 6. - Part selection.

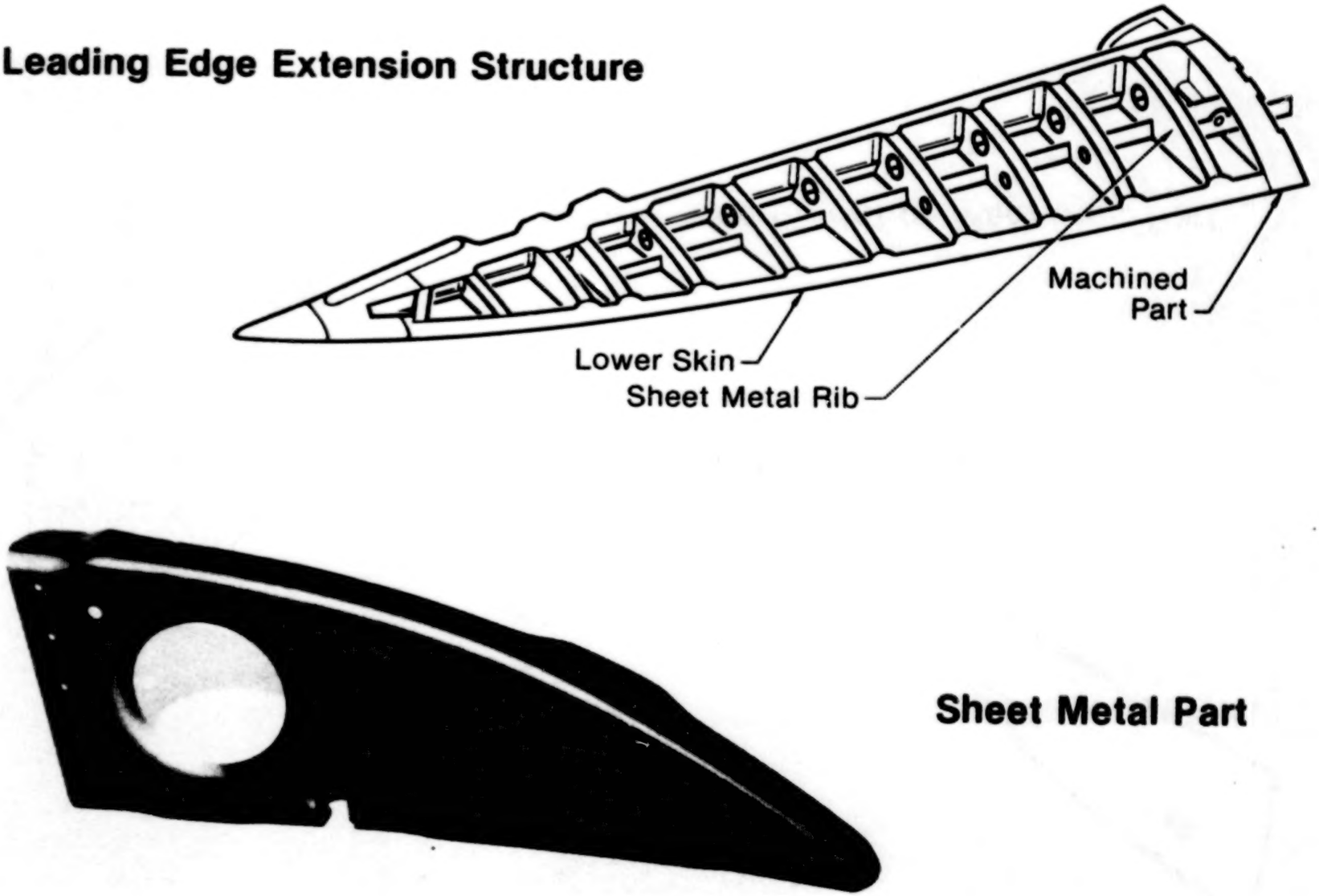


Figure 7. - Sheet metal part.

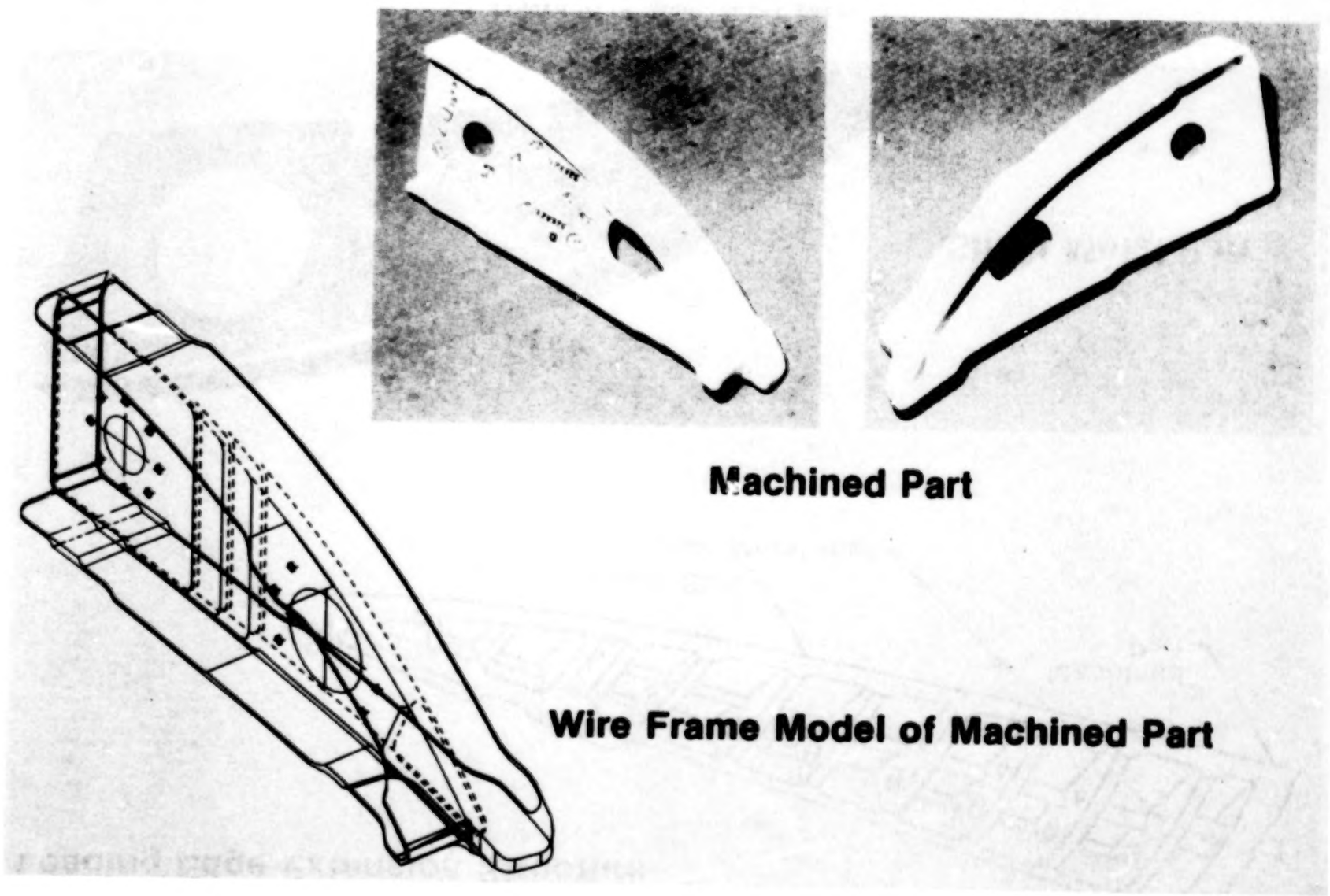


Figure 8. - Machined part.

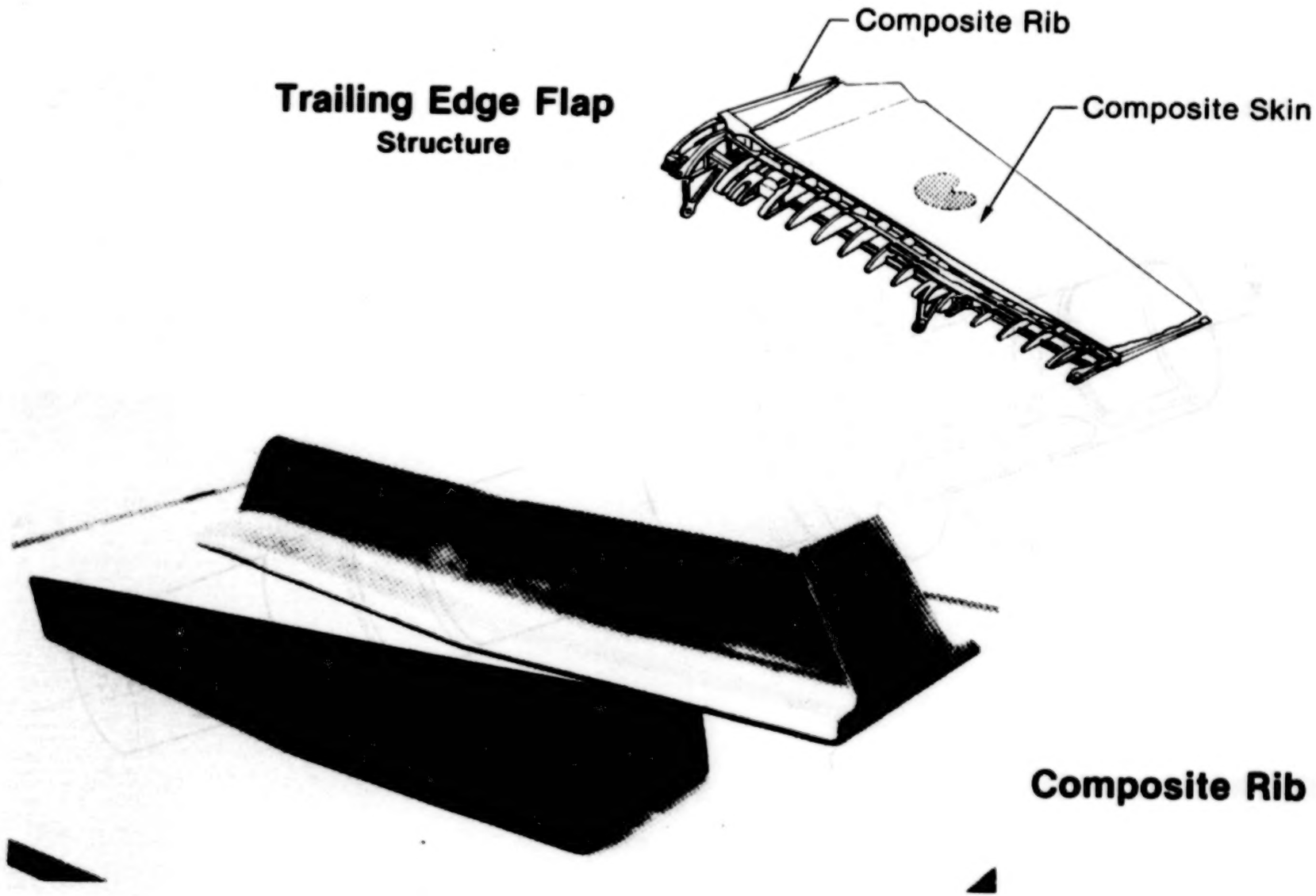


Figure 9. - Composite rib.

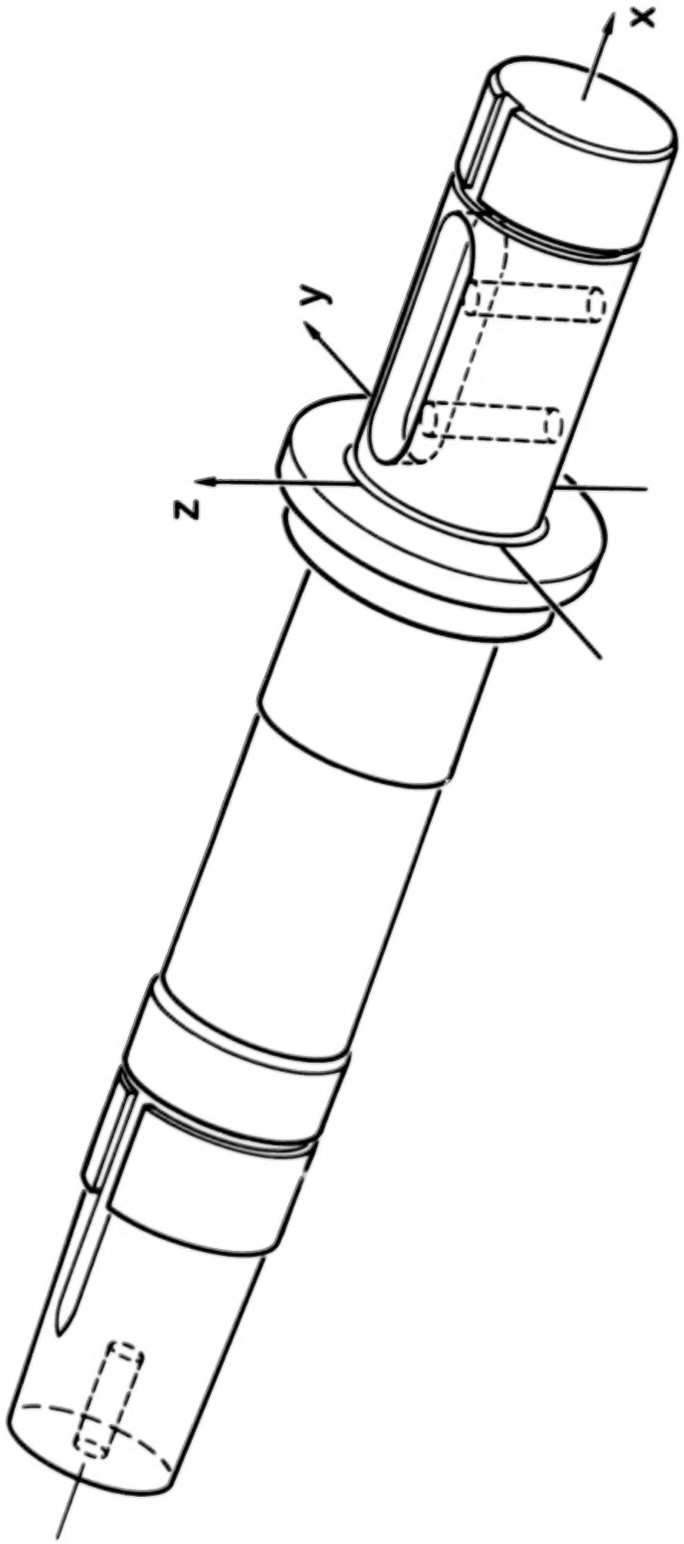
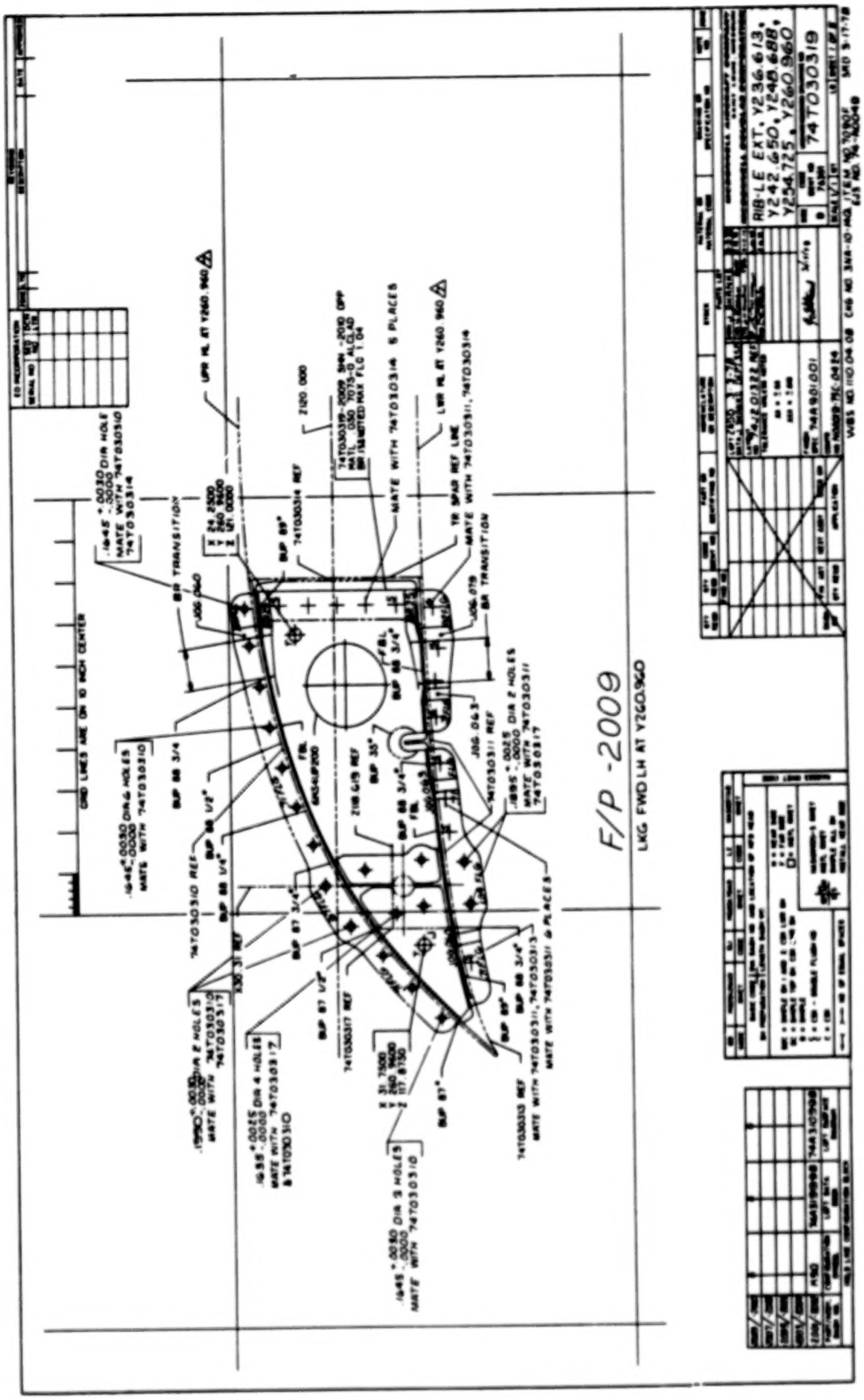
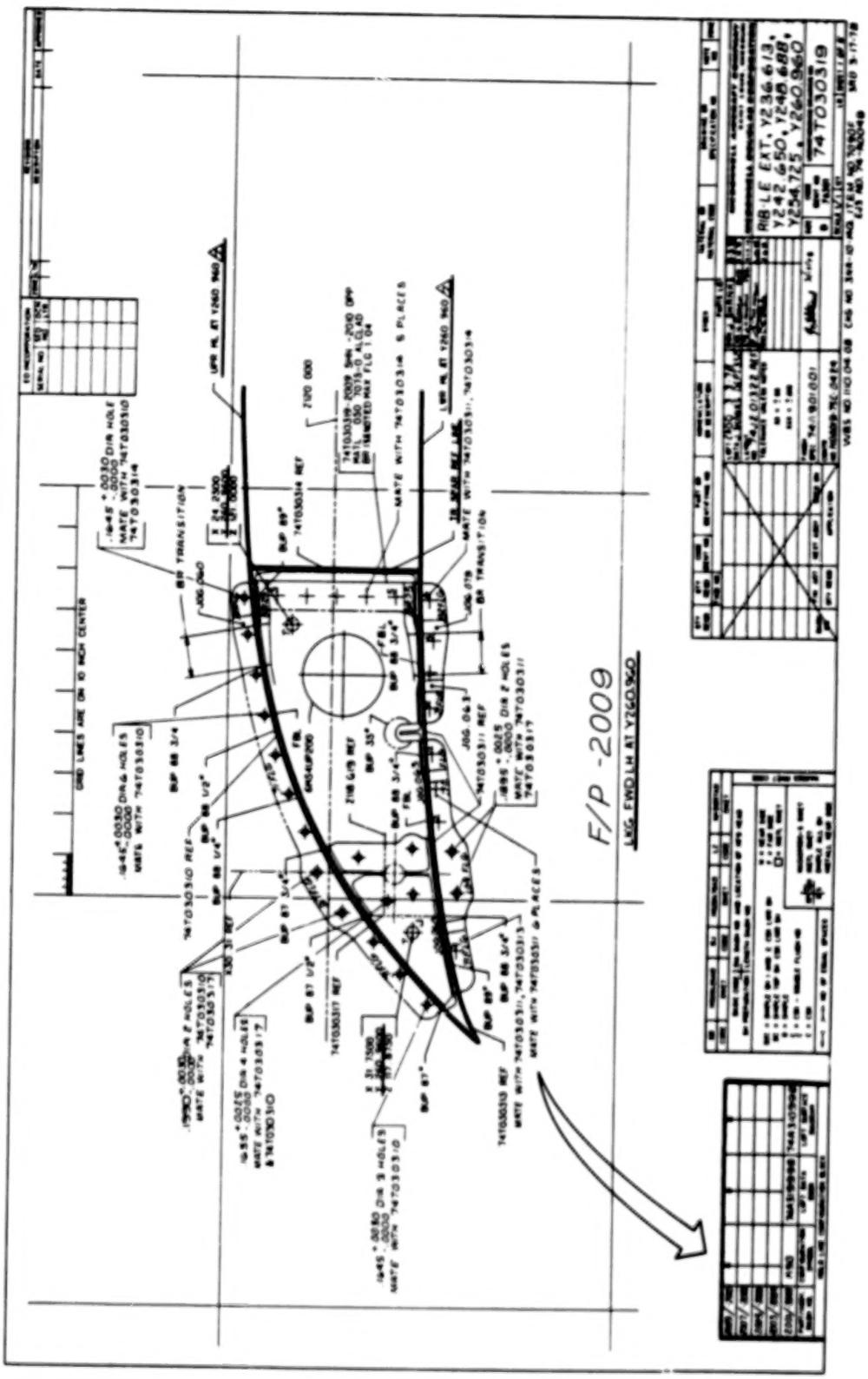


Figure 10. - Turned part.



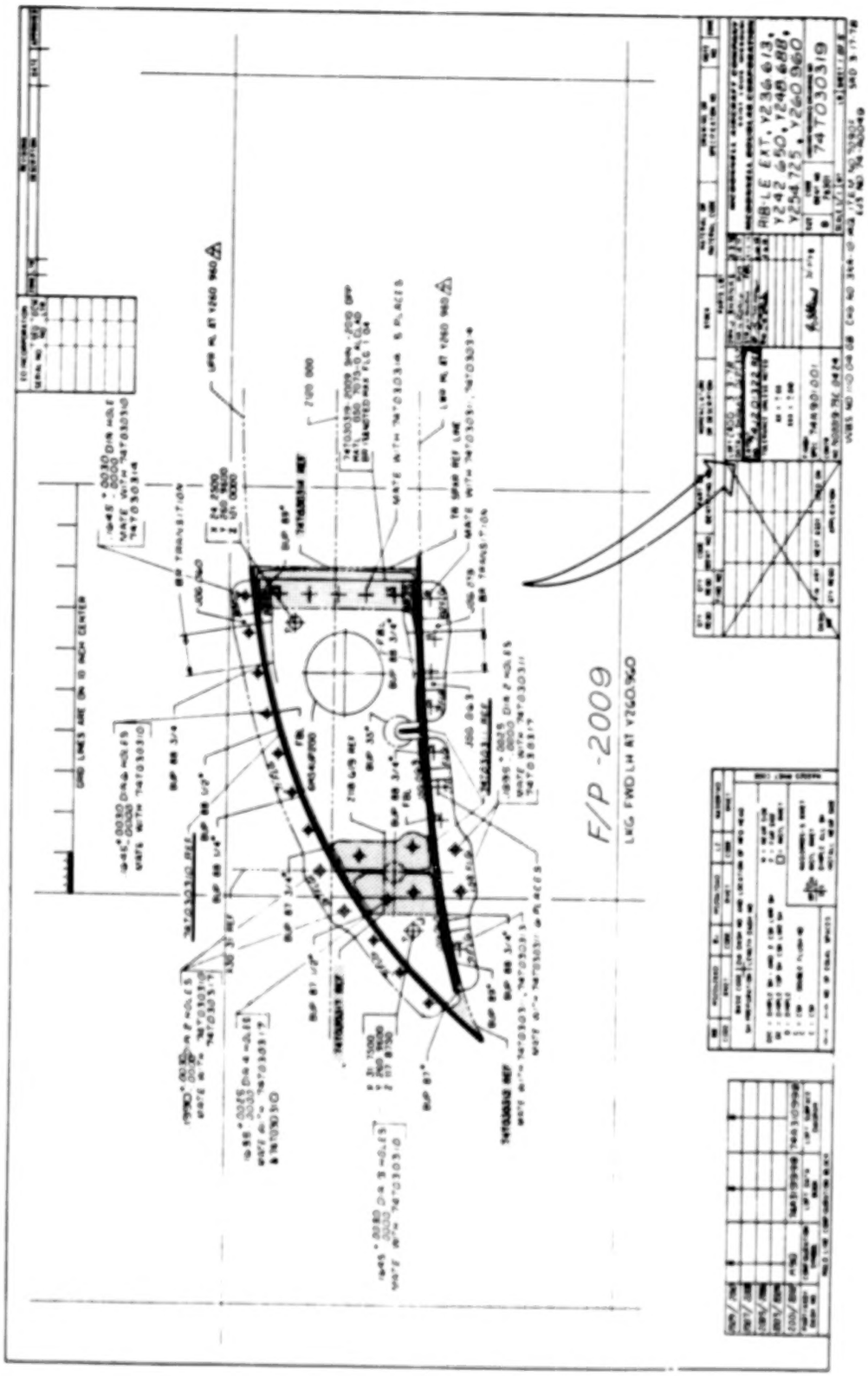
(a) Sheet metal part.

Figure 11. - Conventional engineering drawing.



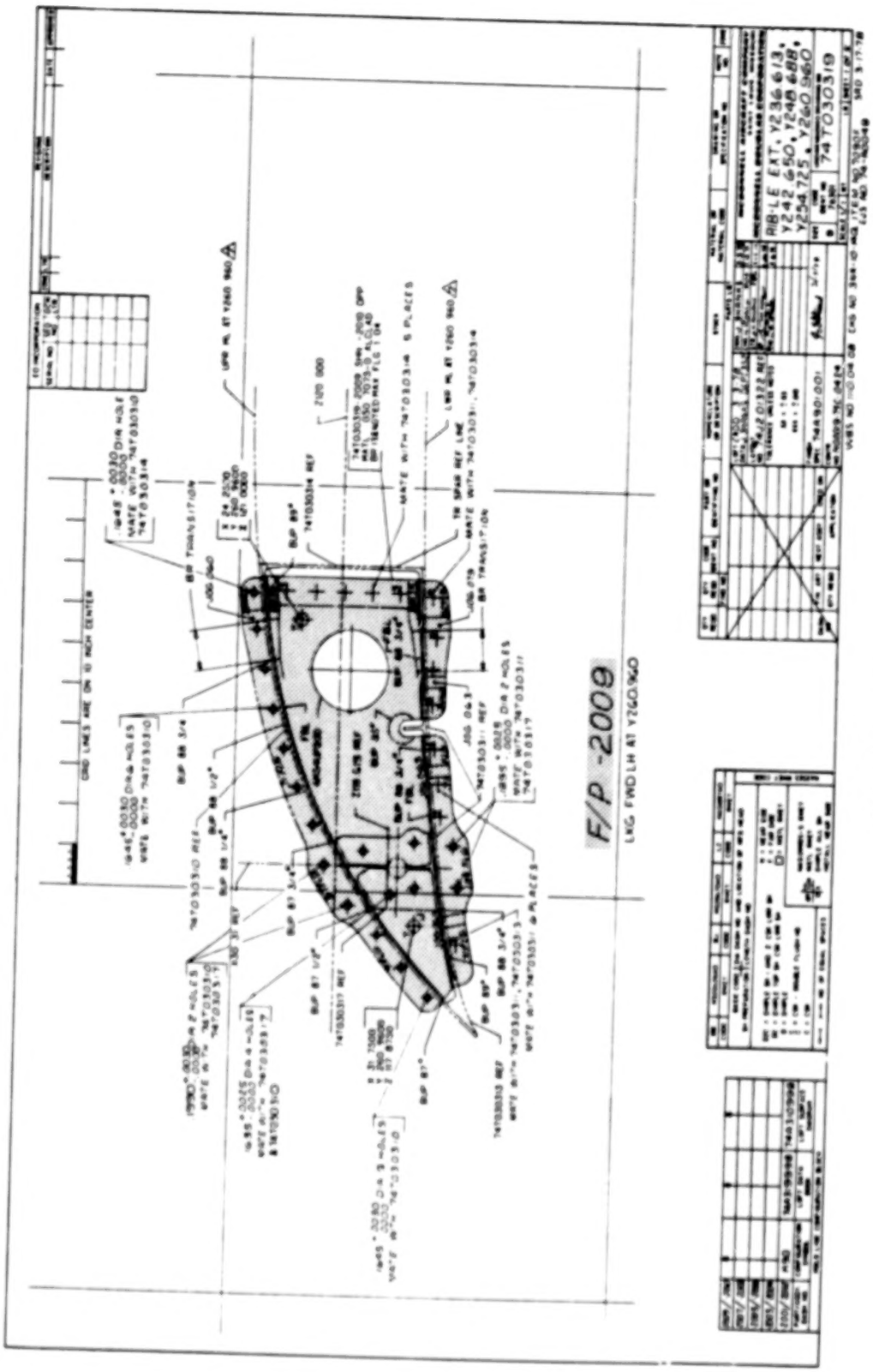
(b) Loft and design data.

Figure 11. - Continued.



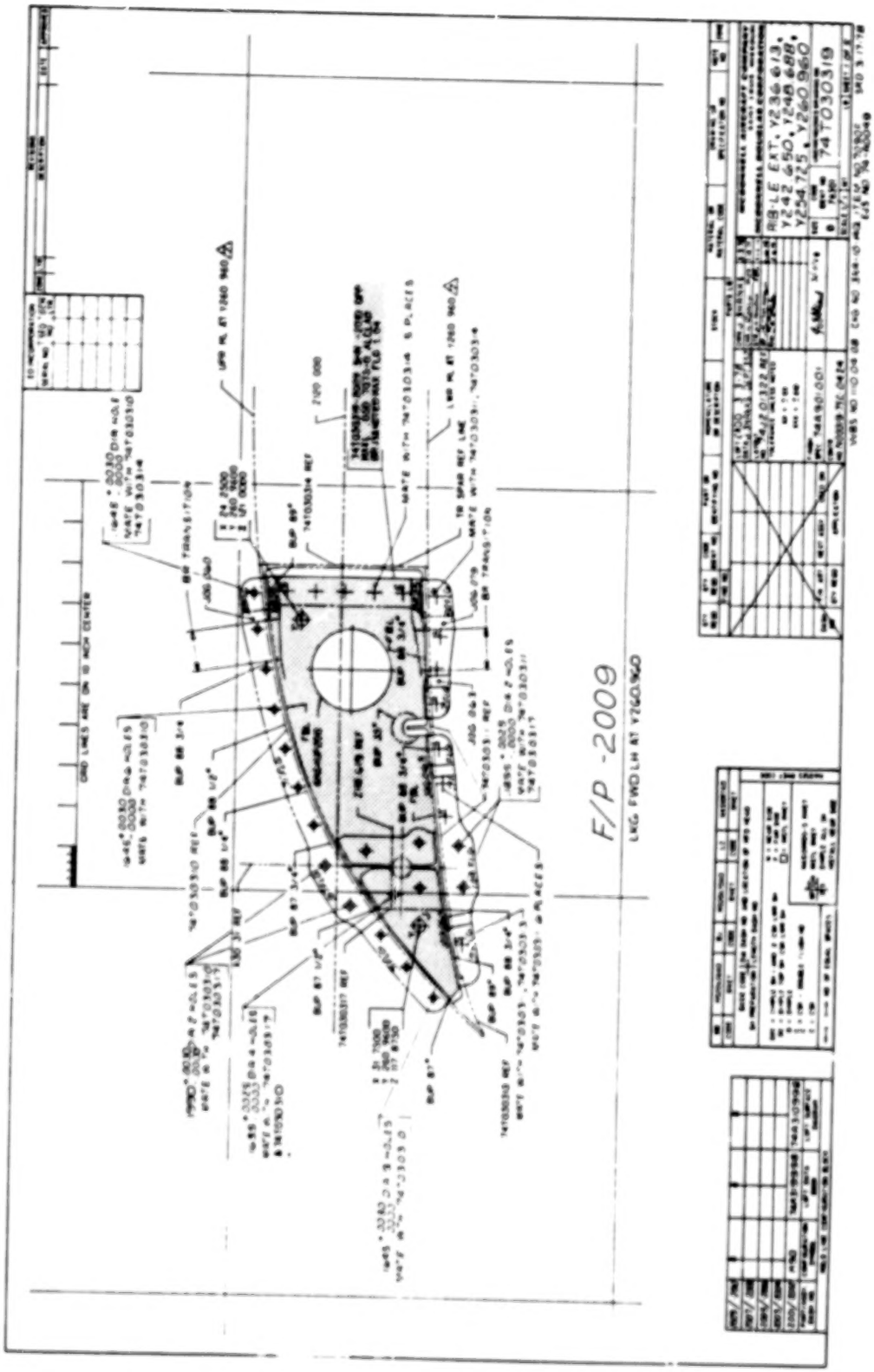
(c) Layout and reference part data.

Figure 11. - Continued.



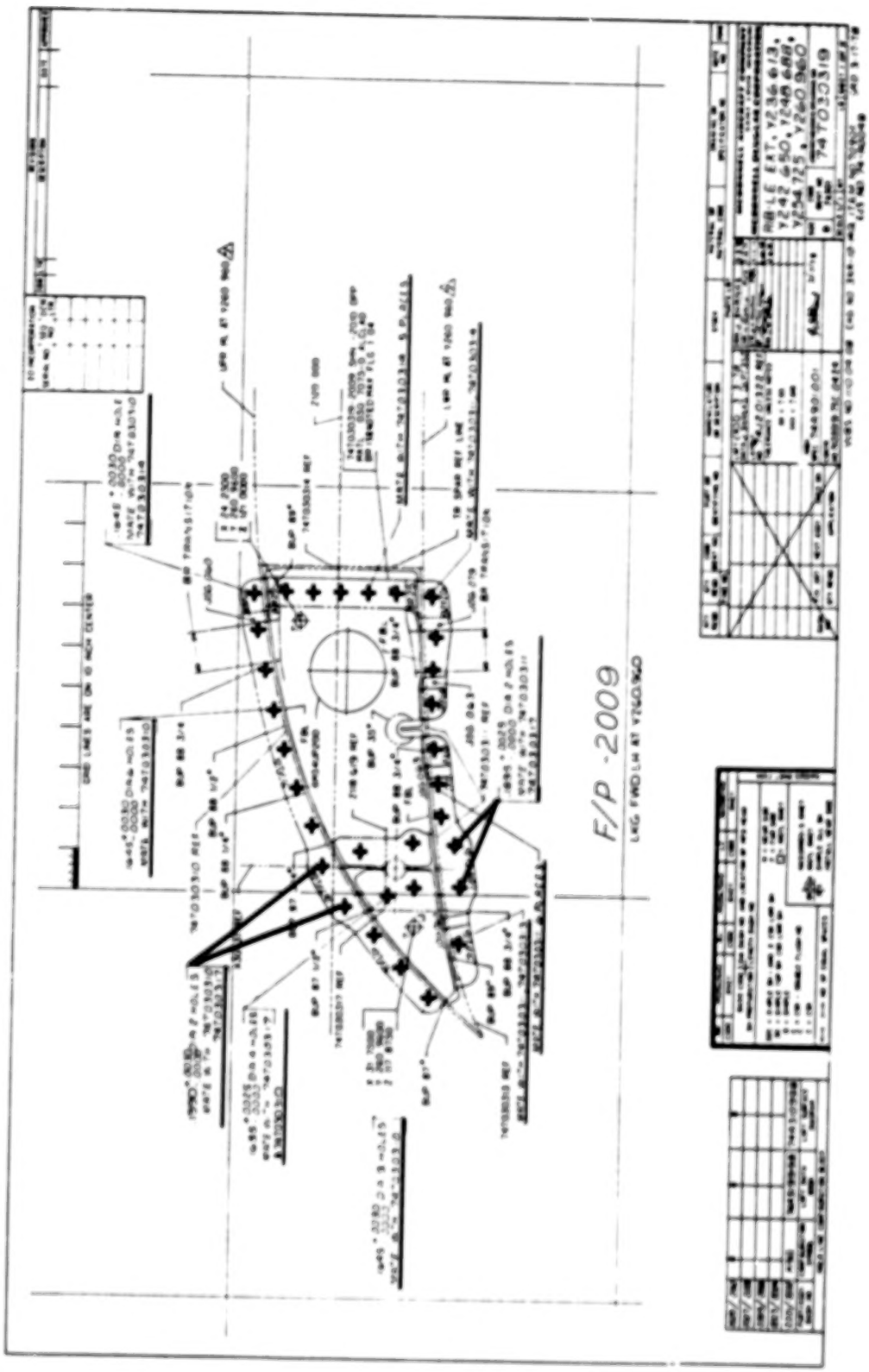
(d) Flat pattern.

Figure 11. - Continued.



(e) Formed part.

Figure 11. - Continued.

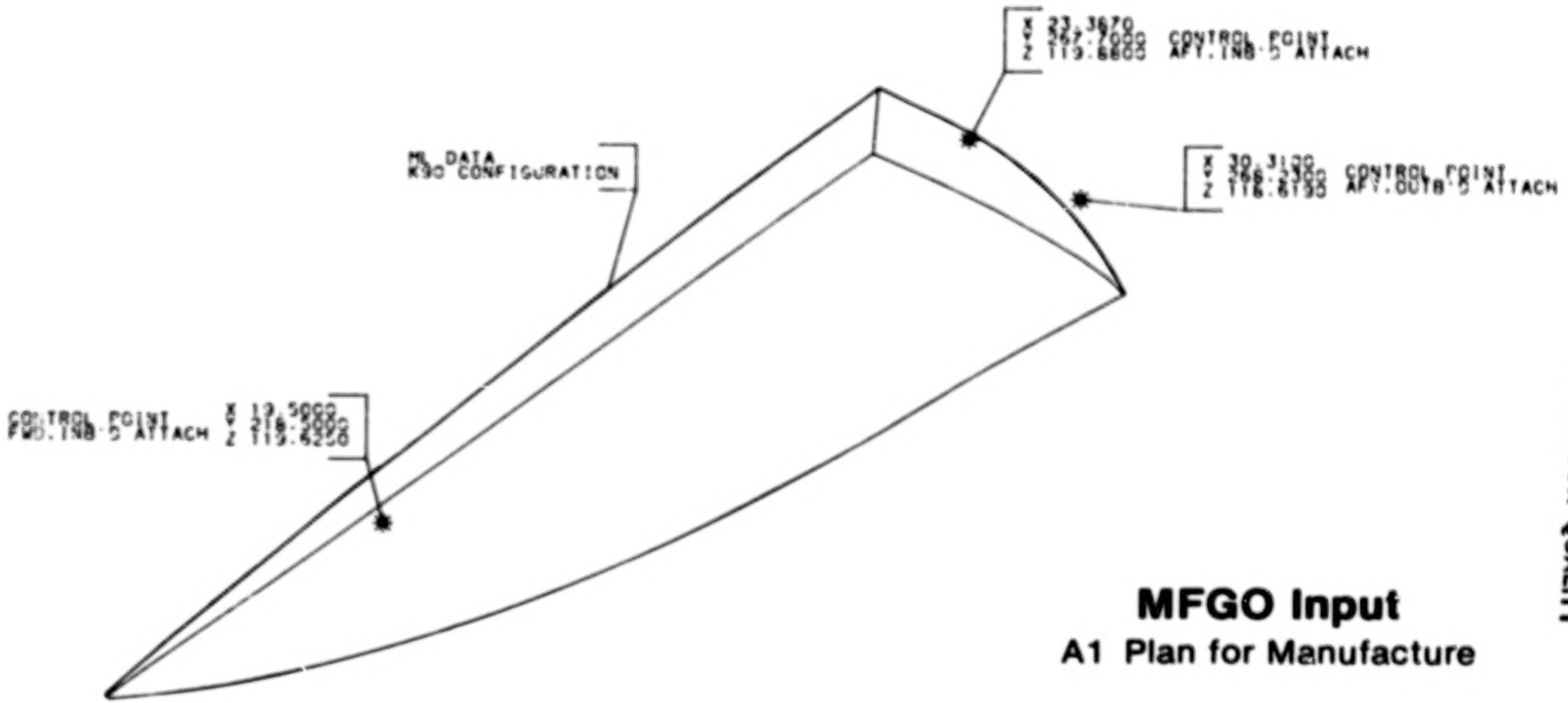


(f) Hole mate with data.

Figure 11. - Concluded.

PDDI Assembly Envelope

— Attach Requirements



MFGO Input

A1 Plan for Manufacture

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 12. - Conceptual analysis - envelope definition.

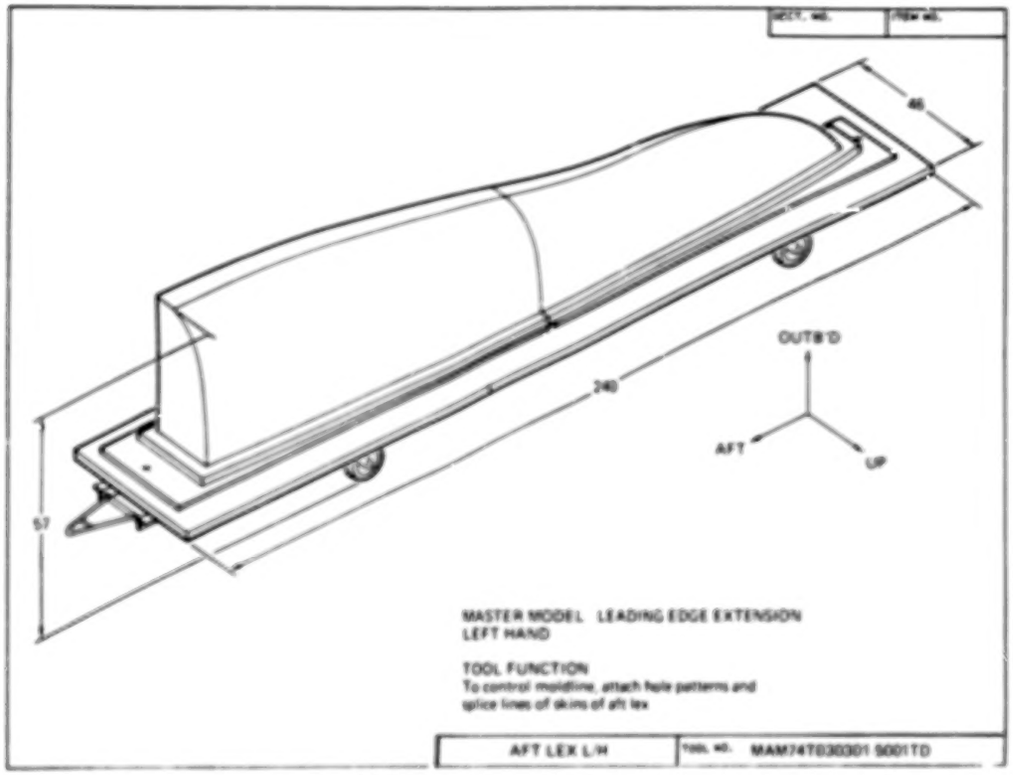


Figure 13. - Master model tooling drawing.

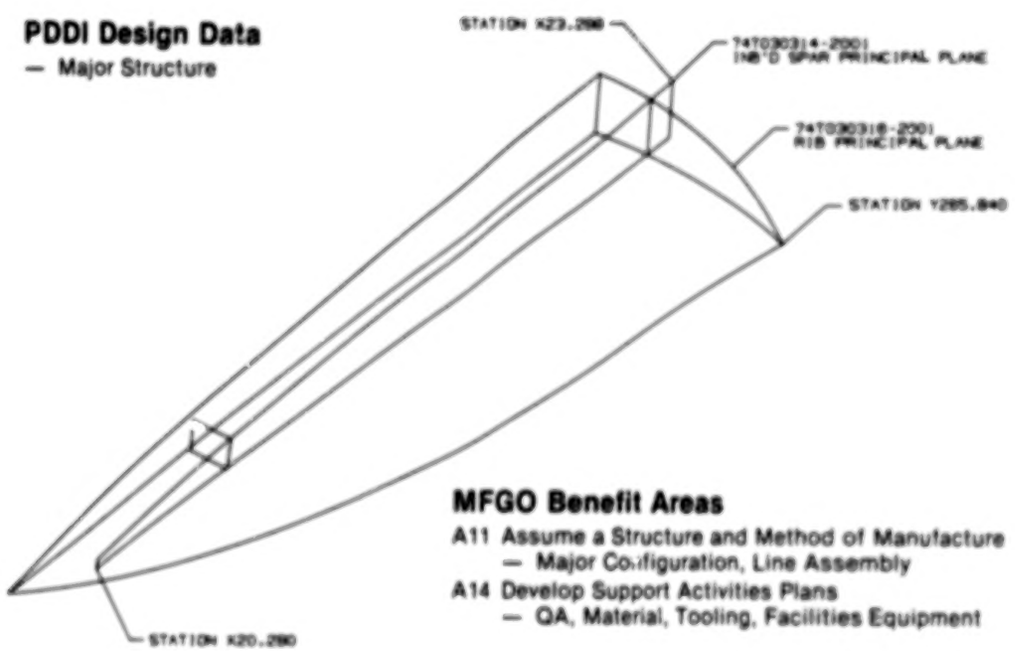


Figure 14. - Conceptual analysis - major structural definition.

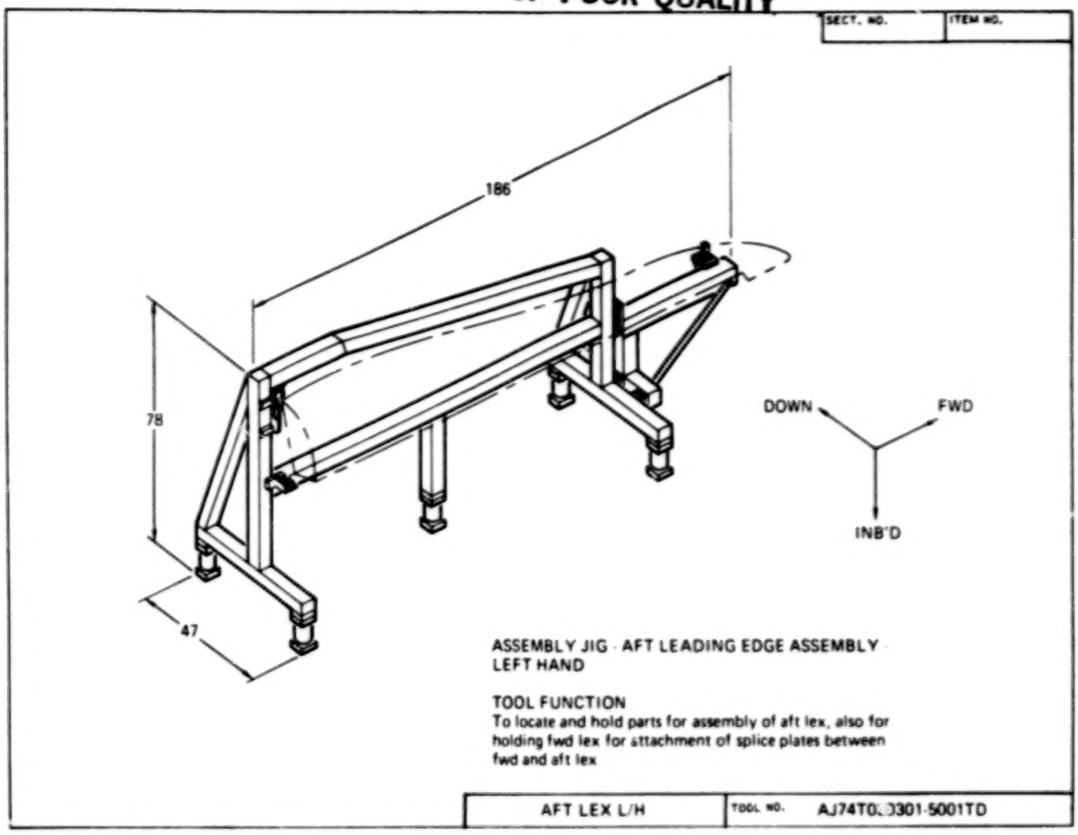


Figure 15. - Assembly jig tooling drawing.

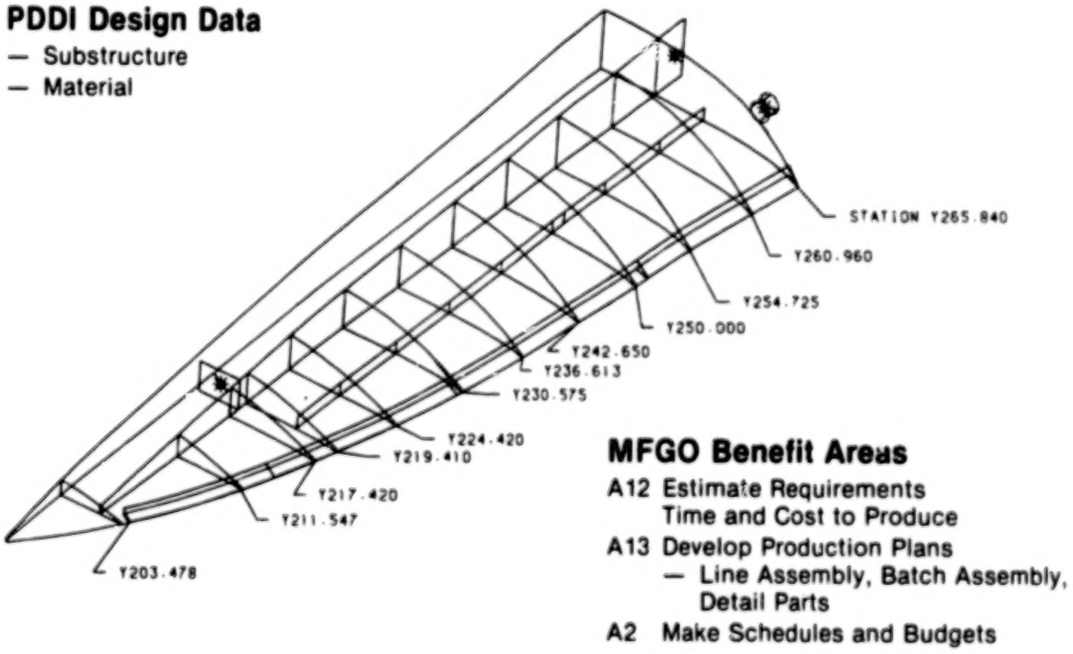


Figure 16. - Conceptual analysis - substructure definition.

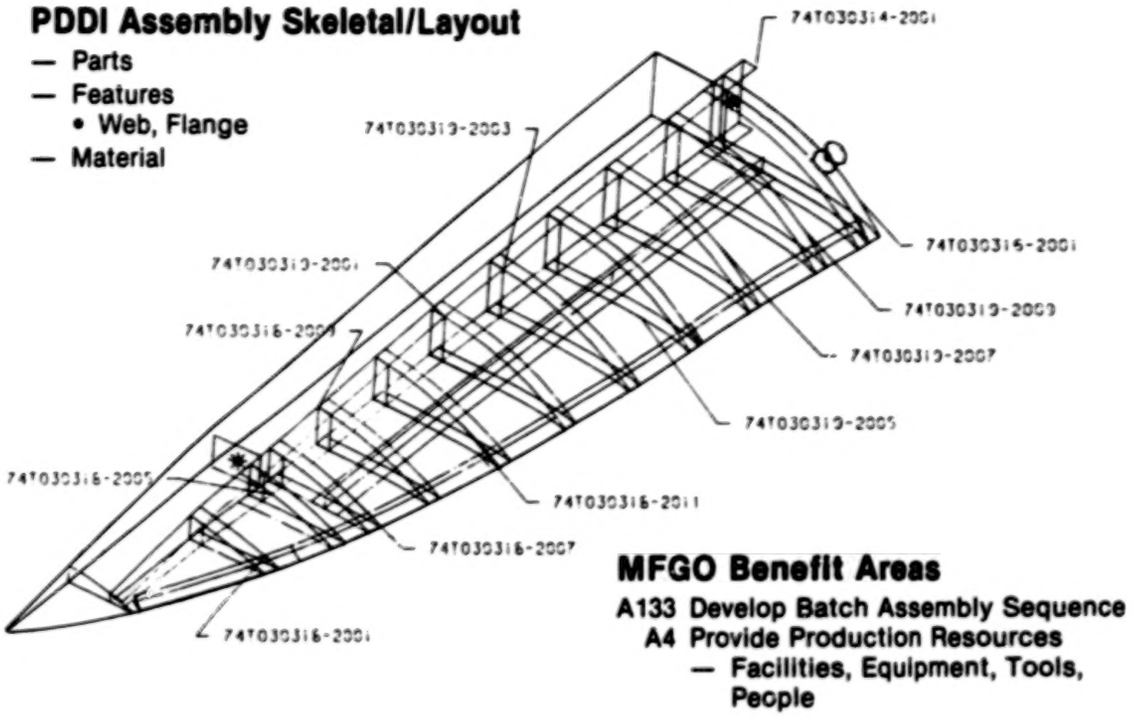


Figure 17. - Conceptual analysis for assembly layout.

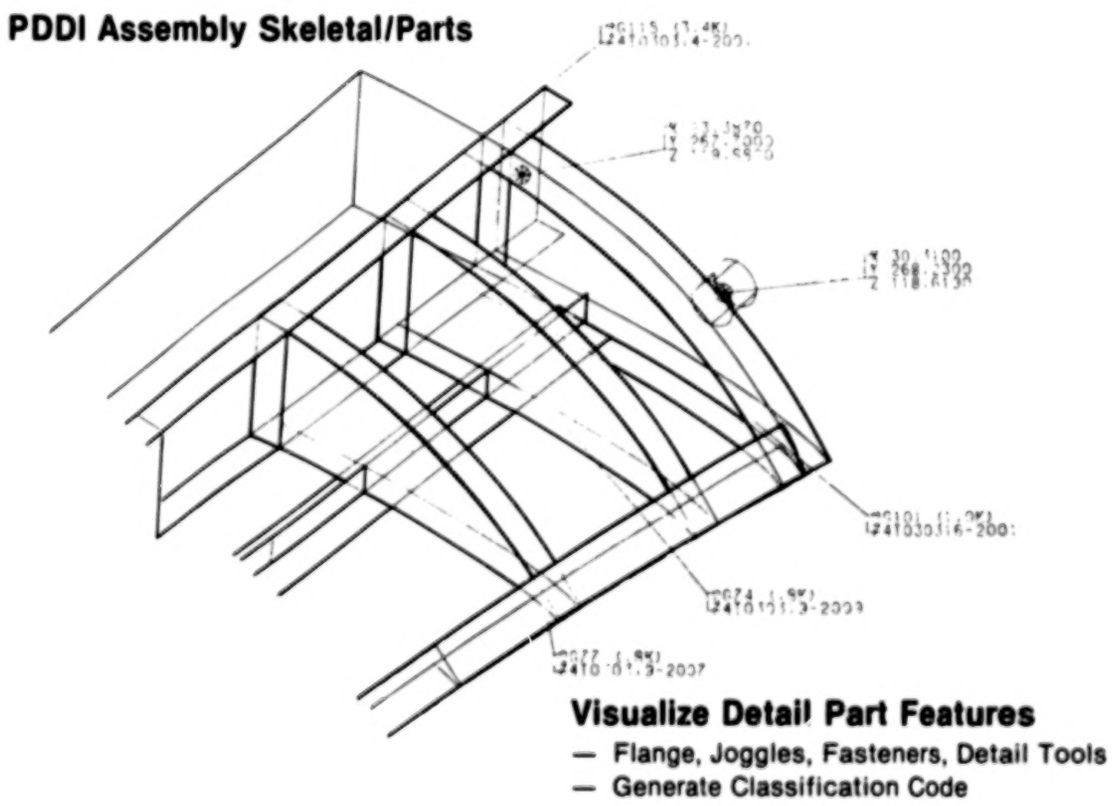


Figure 18. - Conceptual analysis for assembly parts.

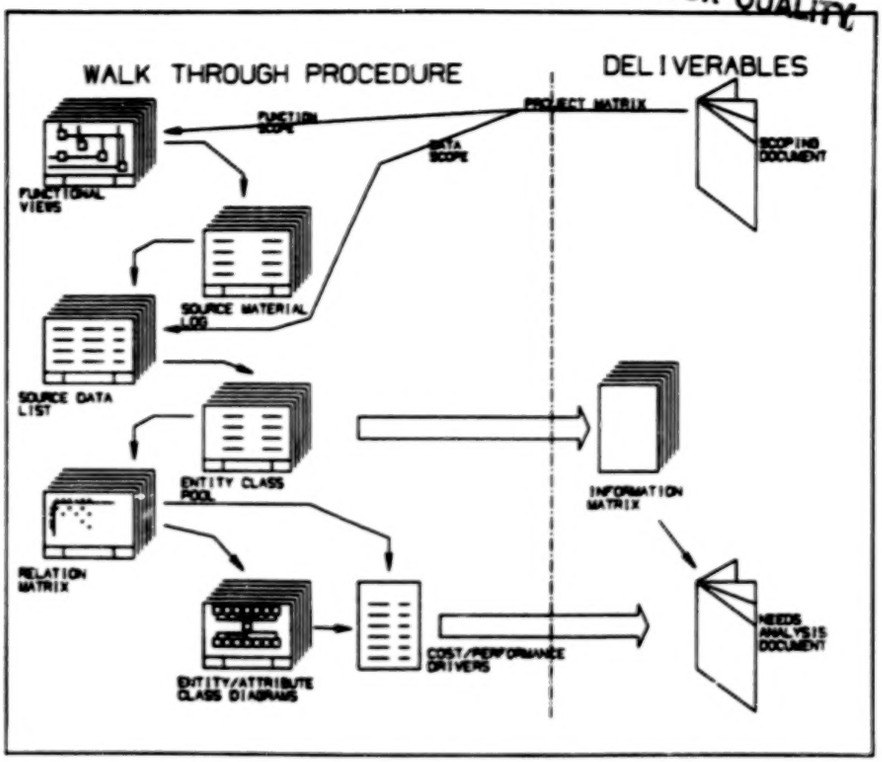


Figure 19. - PDDI needs analysis.

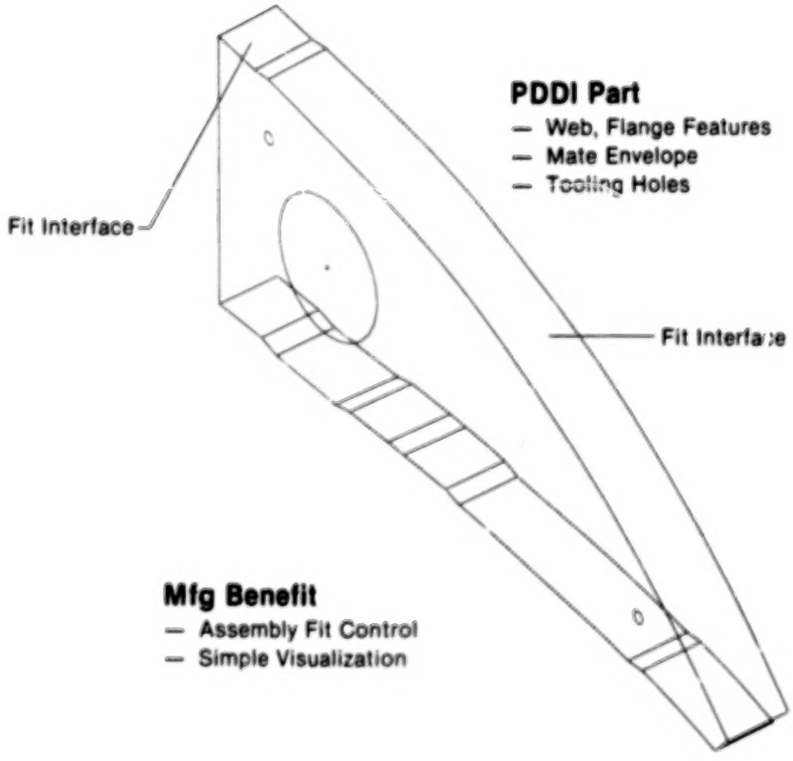


Figure 20. - Detailed analysis - part interface.

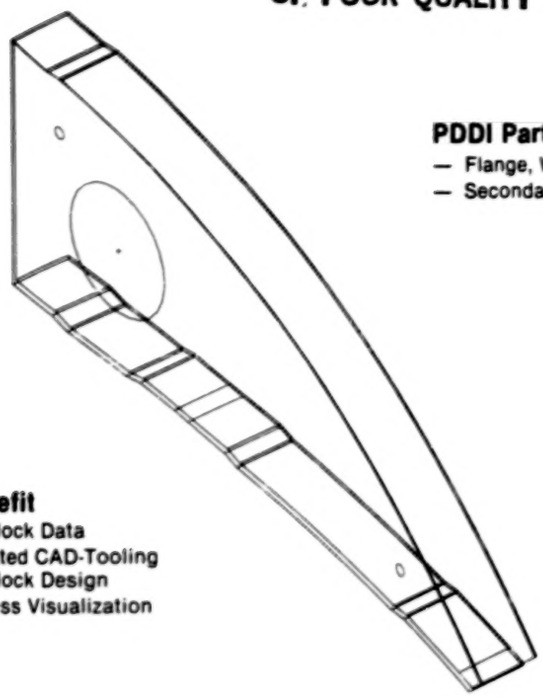


Figure 21. - Detailed analysis - part substructure.

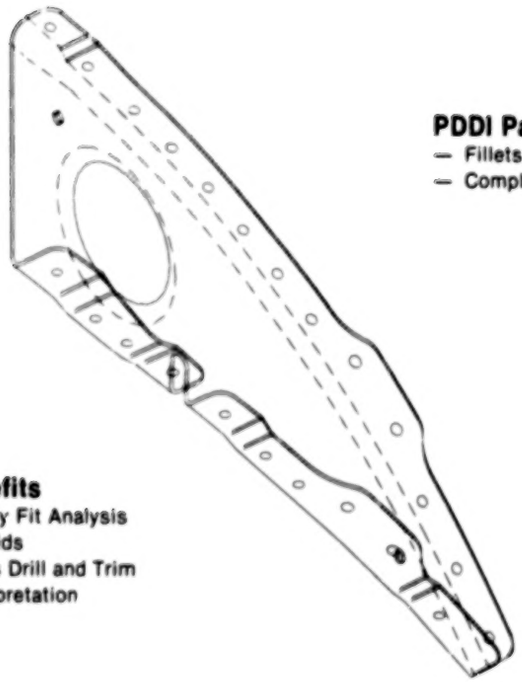


Figure 22. - Detailed analysis - complete part.

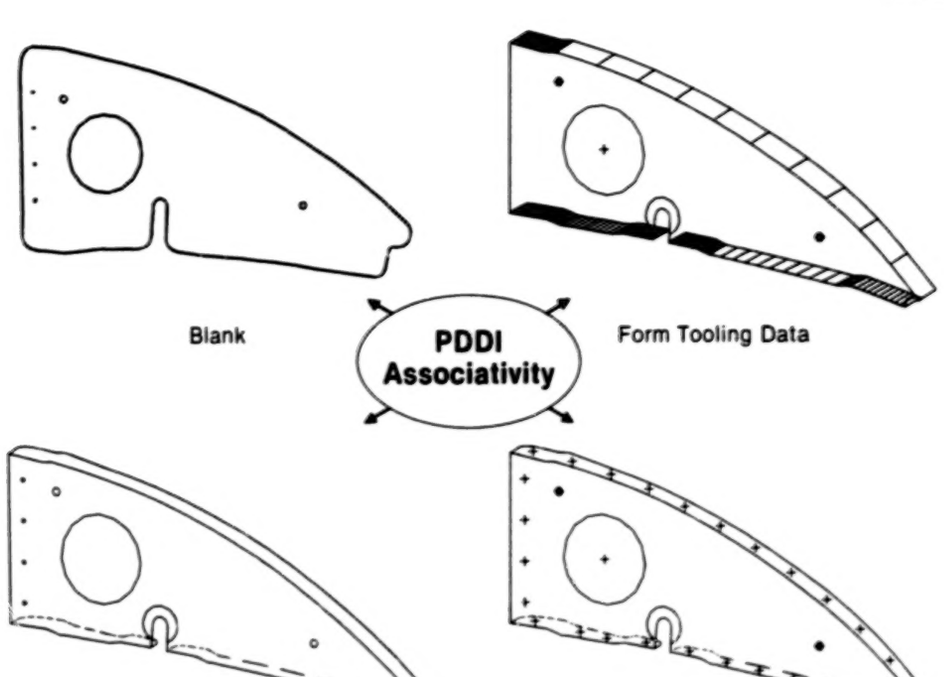


Figure 23. - PDDI data associativity.

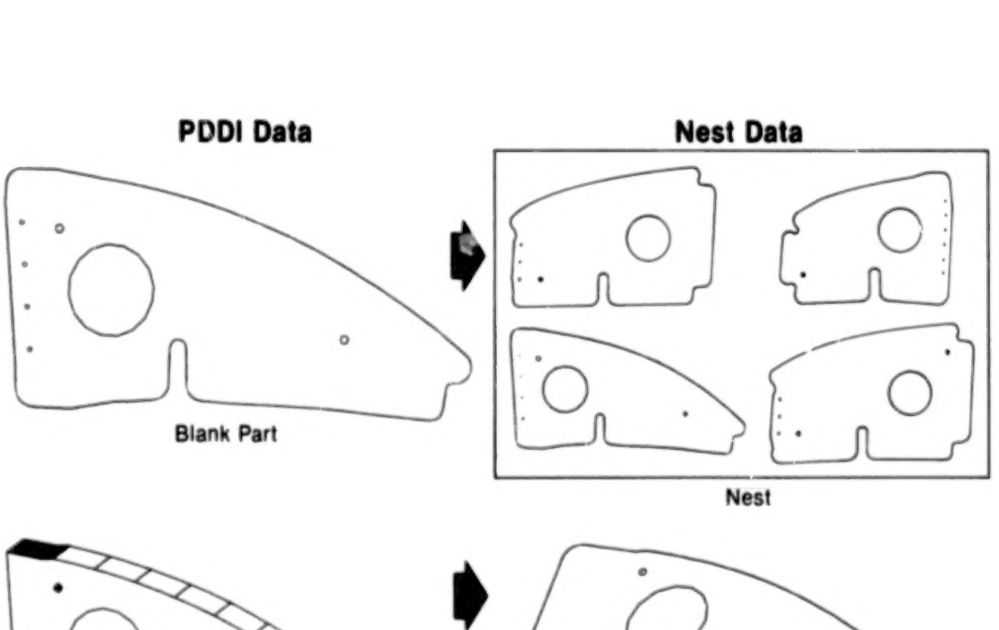


Figure 24. - PDDI interfaces.

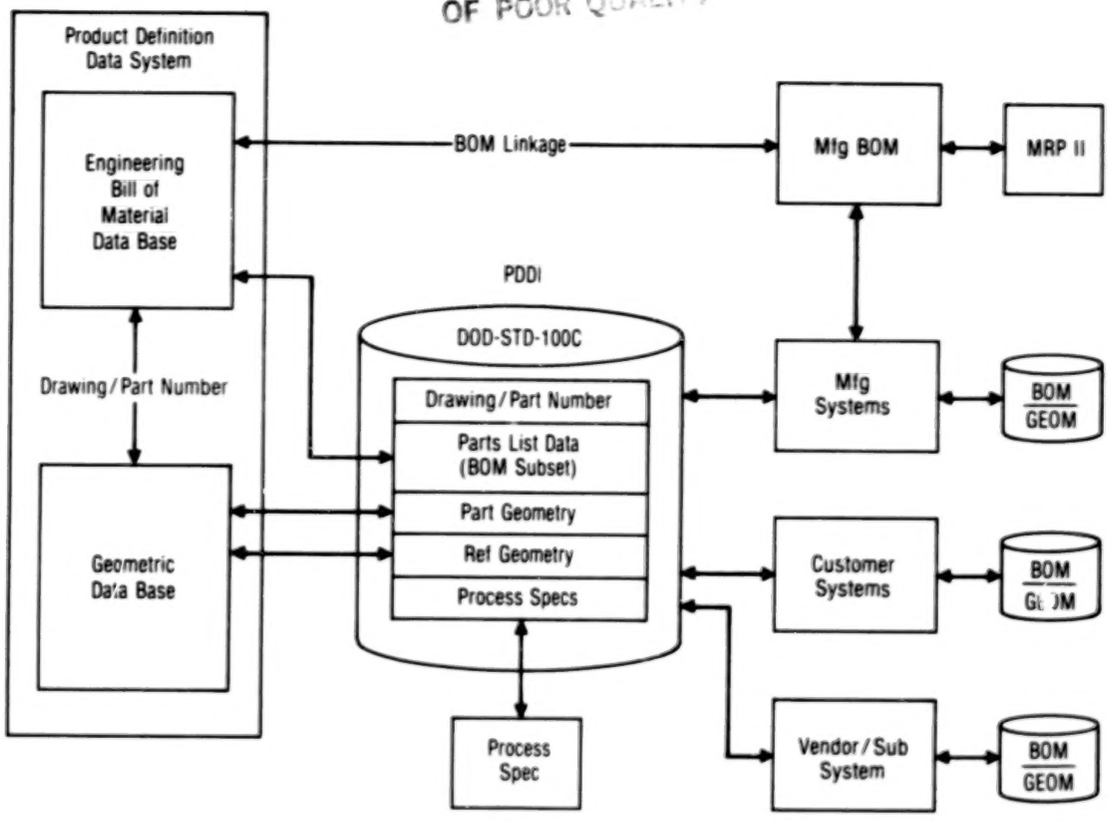


Figure 25. - PDDI - replaces the function of the engineering drawing.

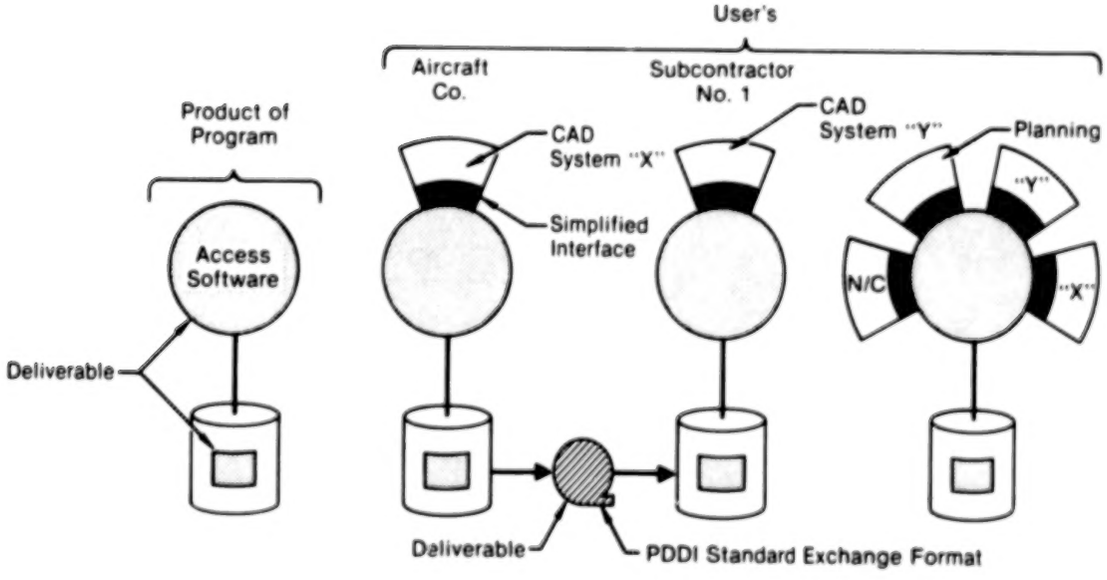


Figure 26. - PDDI deliverable product and its possible use.

DATA DISTRIBUTION IN THE NBS AUTOMATED MANUFACTURING RESEARCH FACILITY

Mary J. Mitchell and Edward J. Barkmeyer
Center for Manufacturing Engineering
National Bureau of Standards
Washington, DC

1.0 Introduction

The Automated Manufacturing Research Facility (AMRF) at the National Bureau of Standards is being constructed as a testbed for research in the automation of small batch manufacturing, in particular for systems producing machined parts in lots of 1000 or less. Construction started in late 1981 and by late 1986 the testbed will be made available for selected research by academic and industrial organizations, research institutions and government agencies. The project is funded by NBS and the Navy Manufacturing Technology Program and is significantly supported by industry through donations or loans of major components and through cooperative research programs. The objects of the program are to identify and exercise potential standard interfaces between existing and future components of small-batch manufacturing systems and to provide a laboratory for the development of factory floor metrology in an automated environment, delivering proven measurement techniques and standard reference materials to American industry. Commercially available products are being used to construct the facility wherever possible, in order to expedite transfer of research results into the private sector.

To provide a real testbed for interface standards, the AMRF is intentionally composed of manufacturing and computing equipment from many vendors, thereby making its construction a major integration effort [Ref. 1]. The configuration is structured around single self-contained workstations, each capable of a well-defined set of manufacturing functions. Each workstation must be capable of operating either as an independent manufacturing unit under control of a local operator or as an element of a multi-workstation manufacturing task under control of a higher-level process. The typical workstation consists of a programmable machine tool, a robot, a materials transfer station, and local buffer areas for tools and workpieces. The 1986 AMRF is expected to contain the following workstations:

(1) Horizontal Milling Workstation
(2) Vertical Milling Workstation
(3) Automated Turning Workstation
(4) Automated Inspection Workstation
(5) Fixture/Tool Assembly Workstation
(6) Cleaning and Deburring Workstation
(7) Materials Inventory and Transfer Systems

The intelligence complex of the workstation includes the robot control system, the machine tool control system, sophisticated sensor systems and a workstation control system to coordinate the activities. Above the workstation level, batch manufacturing coordinators, called Cells,

and a floor manager, called the Shop controller, provide higher levels of control. All of these control and sensory processes are software systems, which reside on a complex of interconnected computer systems, making the AMRF a distributed computing network [Ref. 2]. Each computer system, or "node", in the network supports a group of logically related applications (control and sensory) processes and provides a group of "systems" processes for data management, communications and program control.

A major demonstration of this facility was held at the end of the first stage integration effort in November 1983. At that time, the Horizontal Milling Station, the Automated Turning Station and a Materials Handling System using a single automatically guided vehicle (a "robot cart") were functioning under control of a coordinating Cell. Figure 1 shows the configuration of the 1983 AMRF. The intelligence complex included a VAX-11/780*, two multibus-based multiprocessor systems functioning as workstation controllers, an NBS real-time control system controlling one robot, a locally developed front-end control system for the horizontal machining center, and the manufacturers' controllers for the major equipment components. The physical communications network was a collection of 9600-bps RS232C serial links, using a subset of the ANSI/ISO high-level data link control procedure (HDLC) [Ref. 3].

This paper describes the distributed data management systems used in the AMRF for the first stage integration and discusses the general directions of future enhancement.

2.0 Rationale

For the AMRF to remain a viable research facility for the next decade, the facility must exhibit greater flexibility than existing industrial "flexible manufacturing systems" [Ref. 4]. It must be capable of being readily reconfigured from an integrated automated facility into sets of independent workstations. It must accept addition, deletion and substitution of equipment at all levels. It must accommodate manufacturing equipment and computer systems from many vendors, requiring compliance with accepted industrial standards only. These requirements dictate a highly modular design, isolating individual control or sensory processes by function. Each control or sensory process is associated with a logical identification, which conveys a clearly defined set of functions far more than a particular location or piece of equipment.

For the entire facility to operate automatically, each subsystem must detect and react to failures, discrepancies, uncertainties and unexpected events, which will inevitably arise in the work environment. Systems must therefore be capable of adaptive control, based on sensory and performance feedback. In order to solve the

---

*Certain commercial equipment and software is identified in this paper in order to adequately specify the experimental facility. Such identification does not imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the equipment or software is necessarily the best available for the purpose.

otherwise unmanageable control problems resulting from feedback response systems on such a wide variety of equipment, the control architecture selected for the AMRF is the hierarchical control model defined by Albus and Barbera [Ref. 5]. This model mandates modular design and functional responsibility of control systems and imposes a chain of command structure on the relationship between control systems.

Data flows between control systems in a number of ways. Command and status information flows down and up the logical hierarchy between supervisors and subordinates, while information such as workpiece location and geometry is used and modified by materials handling, robot control and fixturing control processes all over the facility as the workpiece changes hands.

In order to accommodate all necessary data flow, it is necessary to treat the data resources of the AMRF as one integrated database to which all control and sensory processes have access. Each process exercises access only to those information units it needs, while the authority to modify particular information units is limited to particular processes, as in any integrated database management system. The alternative of constructing separate process-relevant databases results in replication of data, concurrency control problems, and continuous reorganization of the databases as new processes are added to the facility, a phenomenon amply demonstrated by corporate databases of the 1960's. The alternative of forcing data exchange paths to follow the control hierarchy results in numerous proxy retrievals - supervisors retrieving information needed only by a subordinate - with a great expansion in the required bandwidths of the data paths and substantial delays in the data transfers.

Independent operation of control systems implies that each control system must have, or be capable of having, all its required data resident locally. On the other hand, integrated operation means that control systems must be capable of acquiring needed information from non-local sources. In order that the control system need not be rewritten to change the mode of operation, the structure of the required data and the method of access to the data must not depend on the residence of the data from the point of view of the control process.

3.0 The Data Administration System

The Data Administration System (DAS) provides a uniform method of access to data for all AMRF processes. Conceptually, all information exchanged among AMRF processes passes through a single integrated database, managed by a single global data administration system. Any process having information which may be of value to another process "stores" that information into a specific "relation" in the AMRF-wide database, and any process needing that information "retrieves" it from that relation.

The conceptual AMRF database is in reality a collection of separate disk-resident databases and memory areas distributed over the subsystems of the facility, and the data administration system is a

distributed set of processes sharing the global data management function. These processes have in common both the interface to the control and sensory processes, seen by the DAS as the "user" interface, and the interface to each other, seen by the DAS as the "network" interface. Because of the multivendor architecture and differences in the requirements for data manipulation, the distributed data administration system in the AMRF comprises many separate data management systems, front-ended by software facilities implementing the common interfaces. (See Fig. 2.)

A typical Data Management System (DMS) is composed of a data dictionary system, a physical database manager, and a logical view process. The data dictionary system maintains the association between logical datanames and the local physical organization; the physical data manager coordinates and executes the data access operations on the storage media; and the logical view process converts between the physical data organization and the data structures visible to the requesting control process.

The user interface implemented by the DAS front end, analogously, is made up of:

(1) a data definition language, allowing the programmer to define the data structures used by the control processes and the data administrator to define and review the storage schemata which locate the physical data within the facility; and

(2) a data manipulation language, providing syntax and semantics for commands to and responses from the DAS, which enable the user processes to operate on the defined data structures.

The DAS network interface has two related components:

(1) a data locating protocol, which allows the data server at one node to find data resident on another node and map local accesses to it; and

(2) a collection of data transfer protocols, which implement various methods of access to the data.

Because all AMRF data must flow through these interfaces, the DAS must be prepared to handle conventional relations, such as inventory and work-in-progress records, rapidly changing command and status interchanges between control systems, and large bodies of "unstructured" text representing controller programs and part geometries. In the 1983 AMRF, these dissimilar data management functions are provided by three separate underlying data management systems, each with its own data dictionary, physical data manager and logical view process (LVP).

No common data dictionary language or server was constructed for the first integration; each Database Management System (DBMS) is managed by a knowledgeable human administrator for that system. A common data manipulation language was developed and is supported by a single data server process running on the VAX node and front-ending two of the three DBMS's; namely those handling unstructured text and conventional relations.

In the 1983 AMRF, the rapidly changing information - all the command and status interchanges and most sensory information - is handled as memory-resident data by separate shared memory management

systems at each node, cooperating with the networking software to provide an AMRF-wide distributed shared memory. It is important to recognize that these systems are the third form of DBMS in the facility, each carrying its own data dictionary mechanism, its own physical data manipulation mechanism, and a primitive LVP converting the local data representation to the AMRF "standard".

The shared memory system, the VAX data server and the two underlying DBMS taken together constitute the existing Data Administration System.

4.0 The Distributed Shared Memory

4.1 The Shared Memory Mechanisms

Interchange of memory-resident data makes use of explicitly allocated memory areas, called "mailboxes", for each functional information unit. Access is limited to a predefined set of mailboxes, each containing a single logical record which is regularly updated by direct replacement (rewrite). The originating process rewrites the information unit in the designated mailbox whenever it has new information to provide. When the retrieving process resides on the same computer system and has direct access to the same designated memory area, it can "read" the information unit simply by fetching from the common memory (Figure 3). In the 1983 AMRF, this is true of interchanges between control systems residing on the VAX (e.g., Cell and Materials Handling). In other cases, the retrieving process may reside at the same node, but has no direct access to the originator's memory. In this case, a transporting process resident on the node must copy the "mailgram" in the originator's mailbox to a separate mailbox designated to receive that information in the local memory area of the retriever. This is the case with the component control processes of the NBS Robot Control System. When the retriever resides on a physically separate node, a Network Interface Process (NIP) resident on the originator's node uses the local shared memory protocol to read the originator's mailgram and transmits a copy of the mailgram over the AMRF network to the NIP on the retriever's system; the receiving NIP then stores the mailgram into the appropriate mailbox on that system, where it can be read by the retriever using the local protocol there (Figure 4).

In all cases, the control process view of the communication is that it stores into or fetches from a shared memory area, according to some protocol common to all processes on that system. Thus the originating process does not have to know where the retrieving processes are, or even which ones they are, as long as it abides by the local protocol; and the retrieving process does not have to know where or how the information originated. To both processes, only the structure and function of the information are significant. This mechanism encourages the development of standard functional information groups, to be created and consumed by control and sensory processes without regard for the mechanics of interprocess communication.

What is clearly implied by this architecture is the presence of a data dictionary which drives the transport processes within and between

systems. In the 1983 AMRF, the true dictionary exists only on paper and is constructed for the transport processes by human operators.

4.2 Shared Memory Protocols

Because the originator and retriever are nominally asynchronous processes, even when they are resident on the same computer system, a number of problems may arise in the interchange.

First, the originator may be making changes to the common area at exactly the same time that the retriever is interrogating the information there. As a consequence, the retriever could conceivably get inconsistent information - the former value of field A and the current value of field B. To avoid this, some kind of access control to the common areas must be imposed. Some systems use a semaphore associated with each mailbox, while others divide a regular real-time interval into a write-only period and a read-only period for access to the common memory. In the 1983 AMRF, the NBS Robot Control System uses the time division mechanism, the VAX uses simulated time division, and all of the others use semaphores.

Second, the retriever may run faster than the originator, and thus interrogate the mailbox more than once before a new mailgram is delivered. Confusion is avoided if the retriever can distinguish "old" information from "new" information. To accomplish this, each mailgram has a sequence number field, which is incremented by the originator whenever the text of the mailgram in that mailbox is changed. To recognize a new mailgram, the retriever has only to compare the current sequence number with the last one processed. The standard sequence number mechanism is also of great importance to the transport processes, which use the sequence number to determine when copying the mailgram to a retriever's mailbox is necessary.

Third, the originator may run faster than retriever, with the consequence that the originator replaces the mailgram more often than the retriever interrogates it, so that the retriever misses mailgrams. In many cases this is harmless, since the retriever only wants the current information anyway, as from a sensor for example. When it is necessary to assure that the retriever sees every mailgram, as in the transmission of commands from supervisor to subordinate, a form of "flow control" must be used. The subordinate process reports the sequence number of the last command received as a part of its status report to the supervisory process, and the supervisor refrains from issuing a new command until it has received this "acknowledgement" for the preceding command.

Finally, every retriever must appear to access the same original. Any completely qualified function in the AMRF must be provided by at most one process at any given time, so that there is exactly one possible originator of any mailgram identified by function. It follows that at any given node, there is exactly one writer of any mailbox, although there may be many readers, and if the originator of that mailbox is not local, then the Network Interface Process at the node is the local writer.

4.3 Mail Delivery Processes

The actual mail delivery processes - the local transport processes and the network interface processes - are simple table-driven machines.

Figure 5 depicts a local transport process. Its table entries identify the source mailbox, the destination mailbox, the associated semaphores, the mailbox length and the last sequence number. On its cycle, a local delivery process examines its command mailbox and modifies its delivery table as directed, then makes one pass through the delivery table, copying each eligible mailgram from source to destination. The usual practice is to copy from the originator's mailbox into the "shared memory" and from the shared memory to the retrievers' mailboxes, in order to avoid coupling semaphores.

Figure 6 depicts a NIP. NIP tables are similar to local transport tables, except that in each entry, one of the source and destination mailbox identifiers is replaced by a network locator. The NIP cycle consists of examining its command mailbox and modifying its delivery tables as directed, then processing all outbound table entries, copying the contents of any mailbox which has changed into a network packet and routing it to the specified network location, and finally, looking up the table entry for each received packet and depositing the mailgram in the proper local mailbox.

In the 1983 AMRF, the commands to the mail delivery routines to build their tables are issued by a "communications manager" process getting its inputs directly from a human operator, who effectively manages the shared memory data directories on paper.

5.0 The Data Server

5.1 Overview

In the 1983 AMRF there is a single data server providing access to all the disk-resident databases in the facility. These databases are managed by two separate data management systems, residing with the data server on the same physical node - the VAX - and servicing requests from all over the network by means of the integrated shared memory described above.

Modules of the data server perform the following steps:

(1) polling command mailboxes to determine when new requests arrive;
(2) parsing the commands;
(3) constructing a sequence of service requests to the appropriate underlying DBMS;
(4) sending the service requests to the underlying DBMS;
(5) converting and formatting the data to match the required user view;
(6) delivering a response and, as needed, the data to the appropriate user mailbox.

The actual database management is performed by two DBMS's:

(1) a commercial relational database management system - BCS/RIM [Ref. 6], for conventional databases requiring set selection and relational operations, and

(2) a special purpose utility for managing large (5000 bytes and up) binary image files containing control programs for robots, machine tools, etc.

The data server is implemented primarily in Fortran and high-level control over the modules performing data server functions is implemented in the NBS-developed Hierarchical Control System Emulator [Ref. 7]. It is this top-level module which provides the control needed for sharing data resources in systems like BCS/RIM which were not designed to support multi-user access. Fortran modules which poll command mailboxes, parse requests, prepare calls to the underlying data management modules and construct response mailboxes are initiated by and run as submodules under the control of the emulator module.

5.2 Modeling The AMRF Data

Where data has shared common elements between control processes but each process has differing access requirements, a common conceptual model of the overall data requirements was constructed, along with a mapping between this conceptual model and the data structures satisfying the individual user requirements (logical view). The common model was normalized into relations and rules were developed to maintain consistency between relations. The normalized relations were then defined as relational tables in the BCS/RIM data definition language. Logically "like" data requirements from different control processes were assigned to a domain which is translated into an attribute in the BCS/RIM representation. This allows for global representation of like elements, while separating occurrences originating from different processes.

The data server implements the AMRF rule that any given unit of information, in this case an occurence or row of a data relation, has at most one authorized writer, although it may have multiple readers. In the first integration, data relationships were known in advance and it was sufficient to limit write access by the control process.

The application requires the ability to store and retrieve binary image data as an entity. Since BCS/RIM does not provide a data type for these structures, a separate primitive DBMS was constructed to manipulate them.

5.3 Nature Of The Data Manipulated

Of immediate concern in the first stage integration is data which is used by more than one component or control process in the AMRF.

Data managed by the data server includes:

(1) Orders and Work Orders: Customer orders for parts fabrication and their status; internal work orders and state of accomplishment.

(2) Kitting Instructions: The work order for removing raw materials from inventory and grouping them for delivery to a workstation prior to fabricating a batch of parts.

(3) Inventory: Count, location and composition of part blanks, workpieces in process and finished parts; allocations of resources for upcoming production orders.

(4) Tray Status: All trays in the material handling system and their current location and status.

(5) Tray Contents: The current contents of all trays in the transport system and the relative workpiece or tool locations in each tray.

(6) Resource Lists: Data resources required to be present at a workstation before production of a particular part batch can be initiated.

(7) NC Programs: Binary image files of numerical control (NC) programs for automated machine tools, with an index which maintains version control.

(8) State Tables: Binary images of programs for workstation controllers in the form of state tables, with an index which maintains version control.

5.4 Control Program Access

Control processes at every node of the network deal with the data server as another control process with a command and status interchange for access to the disk-resident databases. The interface between the control processes and the data administration system is through sets of mailboxes in the distributed shared memory described above. Each process requesting service from the data server establishes three mailboxes:

(1) the command mailbox, where requests for service are written by the control process;
(2) the status mailbox, where status of the last request is written by the data server;
(3) the data mailbox, where the actual data satisfying the request is written by the data server.

The status mailbox provides command acknowledgment as well as sending

affirmative and diagnostic responses. In the first integration, the data server acknowledges a command only when it completes it, and every requestor must wait for an acknowledgment before issuing another request for service.

Because the set of data operations performed by a given process is usually known in advance, arrangements can be made by the data manager to accommodate known processes before the processes begin execution. In the 1983 AMRF, the data server initialized the access paths for all the control processes when it itself was initiated.

Explicit requests for service are made through a common interactive language similar to that provided with BCS/RIM. Several differences are listed below.

(1) The syntax is closer to the ANSC X3H2 baseline document for the relational model [Ref. 8].
(2) Control processes use a simplified syntax for selection lists and value expressions and provide the minimum conditions needed to qualify the set to be operated on.
(3) A request may be directed at a predefined view which is derived from one or more permanent relations.

Requests are decomposed into operations on a single BCS/RIM relation or class of binary data. For relations managed in BCS/RIM, the user-provided arguments are decoded by the parser and converted to BCS/RIM equivalents, which are then passed down to the BCS/RIM interface. Some of the defined service requests trigger multiple actions against the underlying physical database. Transactions were predefined to enable the data server to perform automatically some of the data aggregation functions and to format the data elements, depending on the requesting process. For example, type conversion is provided for control processes requiring display format for data stored as binary integers. Thus, the data server is precoded to build the desired logical views of the physical data, supplying each user process with data in the form specified by the requirements.

5.5 Binary Image Data Utilities

The logical unit of data in a binary image file is typically one very large record which has no meaningful subdivisions. It is delivered and updated as an entity, and the database serves only as a repository. This form is typical of binary program texts, numerical control programs for machine tools and robots, representations of part and workpiece geometries, and machine representations of operations plans and process plans. In the 1983 AMRF, control programs are required to manage these large information units in a single access. The data server maintains the logical records a separate VAX/VMS files, with a BCS/RIM relation providing the index to these files. In addition, the index allows selection of multiple versions of the control files and maintains length and owner attributes.

5.6 System Users

In the 1983 AMRF the data server provides service to thirteen control processes, including:

(1) the operator interface;
(2) the dispatchers, schedulers, and the part batch queue manager in the cell;
(3) two workstation control processes;
(4) two NC machining centers;
(5) the material handling system and its inventory control subsystem.

6.0 Description Of The Next Stage Of Implementation

6.1 Database Systems

The current AMRF databases are a small subset of those anticipated in the facility. An early outgrowth will be the distribution of disk-resident data to several nodes in the network. In addition to the task of providing a common access path, which has already been addressed, the remaining issues are how to distribute the data and how to maintain relationships between data which is physically distributed.

The hierarchical control structure simplifies this task in many cases. Each control process has a defined functionality and defined relationships with other control processes. Analogous segments of the data which are proper to particular control processes can be similarly distributed and the relationships to other data similarly maintained. For the majority of data classes, the entire set will be maintained on a single node. In particular, data sets which are entirely originated by a single control process will be resident on the node with that process. Other classes have occurrences originating from several control processes located on different nodes. The decision whether to locate the class centrally and distribute the accesses or to distribute the data with the inherent concurrency problems will be made on a case-by-case basis. The requirement for independent operation of the workstations encourages distributing the data itself.

In order to discuss concurrency problems in distributed access, we distinguish three activity levels for the data.

(1) "Static" data is written once and referenced by many processes; such data includes control programs, part geometries, bills of materials, etc.
(2) "Dynamic" data is rewritten approximately as often as it is referenced.
(3) The remaining data classes are designated "active", meaning that they are frequently modified but much more frequently referenced.

For static data the concurrency control problem is version control. Distribution of static data often involves having control processes make local copies of the original and therefore involves a control process decision to replace that copy with a newer version. We see this as a control system problem and not as a data administration

problem.

For dynamic data, there exists only one logical copy of each occurrence, and every reader must retrieve the current value before taking any action based on this data. The concurrency problem which arises is the asynchronous control problem: One process may make a decision on the basis of a view of the world in which an important change is imminent. The data administration system cannot control the change; it can only affect the time required for the change to be noticed. If that time is critical, it becomes a requirement on the DAS.

The "active" data classes face the concurrency problems addressed by distributed database research. The approach we are considering is to assign an administrator process for each class of this type. The administrator awards authority to add and modify occurrences and takes responsibility for controlling the consequences. The criterion for selecting the administrator process is the association between the function of the process and the function of the data.

Much of the data belonging to low levels may be viewed by higher levels as domains within larger sets. For example, in the first implementation, the view of the set of NC programs from other than their origin contains an attribute identifying the domain (i.e. the workstation) to which they belong. A critical responsibility of the administrator process for such a set is cognizance of the domains.

6.2 The Distributed Data Dictionary

The second problem is that the data dictionary functions are almost entirely human-administered at this time, and therefore tests and modifications of control programs inevitably require the foreknowledge and cooperation of the database systems programmers. This method of data administration will not be sufficient to perform real distributed management of the data. The development task of the next level AMRF is to automate the data dictionary functions; in other words, to manage the "metadata" - the information about the data. The primary functions to be automated are:

(1) access to information on structure, location, physical mapping and DMS selection for AMRF data elements;

(2) definition of user views and associated tranforms by node, control process, and procedure;

(3) maintenance of dynamic relationships between control programs and databases.

This information is incorporated in online data dictionaries accessible to the DAS at execution time. Each node performing data management will have a Data Dictionary System (DDS) that maintains the metadata on all data resources residing on the node and selected information on commonly used resources on other nodes, descriptions of the data requirements for local processes, and paths to other dictionaries in the DAS.

The development task will entail the contruction of a common data definition language (DDL) and implementation of a set of software

tools providing control programmers and database administrators with the capability of building and modifying database structure descriptors, including those for memory-resident data. To maintain the dynamic relationships, the existing data manipulation language (DML) must be expanded to provide facilities by which control programs initialize and terminate sets of accesses to AMRF databases, and the data servers must be extended to implement this interface to the DDS from local and remote control processes and to establish the necessary paths to the data. This allows dictionary functions to be performed "actively". In addition to the insertion of "static" dictionary information through the DDL, metadata is captured from the control process access initializations and from other data servers during path construction between nodes.

6.3 Communications

By early 1985, the communication hardware will be replaced with a modern local area network, providing logically complete connectivity and much higher data rates. Except for performance, this replacement should not affect the control processes or any of the higher level data distribution protocols.

7.0 Summary

The NBS AMRF exemplifies one approach to integrating a set of heterogeneous distributed databases. While the 1983 implementation is primitive, it provides real-time control processes with access to conventional databases, rapidly changing memory-resident data, and large binary files. It demonstrates the feasibility of front-ending existing data management systems with a Data Administration System implementing a common user interface, constructing process-dependent logical views of the data, and providing networked data access. The major shortcoming of the current implementation is the absence of a common data dictionary, and the major task of the near future is the development and automation of a distributed data dictionary system.

References

1. Simpson, J. A., Hocken, R. J., and Albus, J. S., "The Automated Manufacturing Research Facility at the National Bureau of Standards", Journal of Manufacturing Systems, Volume 1, Number 1, 1982.

2. McLean, C. A., Mitchell, M. J., and Barkmeyer, E. J., "A Computer Architecture for Small Batch Manufacturing", IEEE Spectrum, Volume 20, Number 5, May 1983.

3. "Advanced Data Communications Control Procedures", ANSI X3.66-1978, American National Standards Institute, NY, 1978.

4. Merchant, M. E., "World Trends in Advanded Manufacturing Technology," Proceedings, 9th Annual Tri-Services Manufacturing Technology Conference, Orlando, FL, 1977.

5. Albus, J. S., Barbera, A. J., and Nagel, R. N., "Theory and Practice of Hierarchical Control", Proceedings of the 23rd IEEE Computer Society International Conference, Washington, D.C., 1981.

6. "BCS/RIM - Relational Information Management System Version 6.0", TR 70101-03-017, The Boeing Company, Seattle, WA, May 1983.

7. Milligan, S. D., and Johnson, T. L., "Hierarchical Control System Emulation Programmer's Manual", NBS-GCR-82-414, National Bureau of Standards, October 1982.

8. "Draft Proposed Relational Database Language", ANSC X3H2-83-49, American National Standards Committee, ANSI, February 1983.

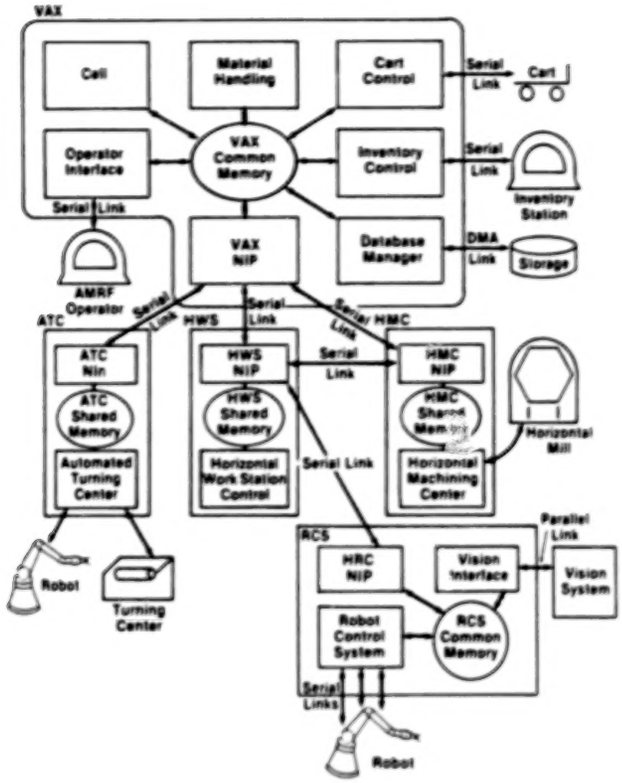


Figure 1.- The 1983 AMRF network topology.

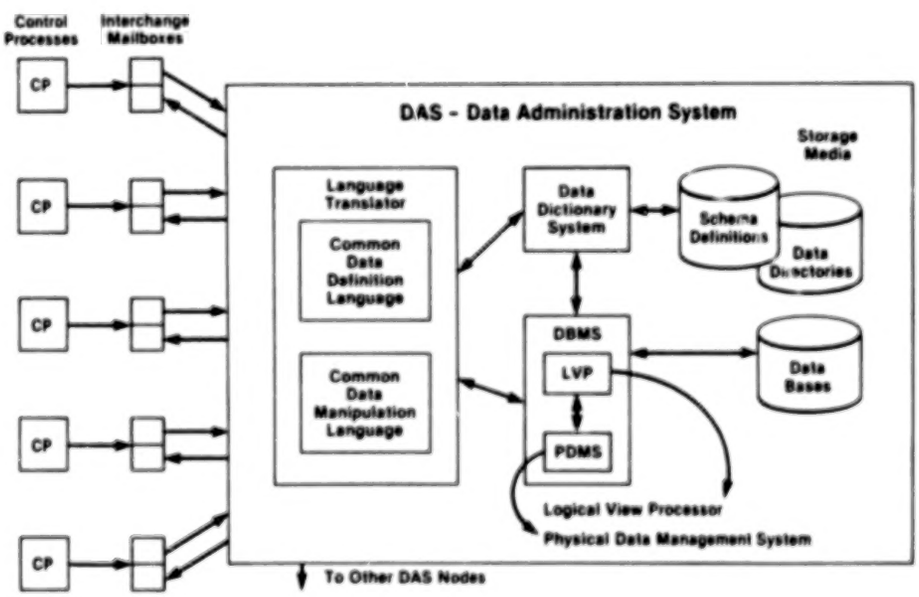


Figure 2.- Data administration system.

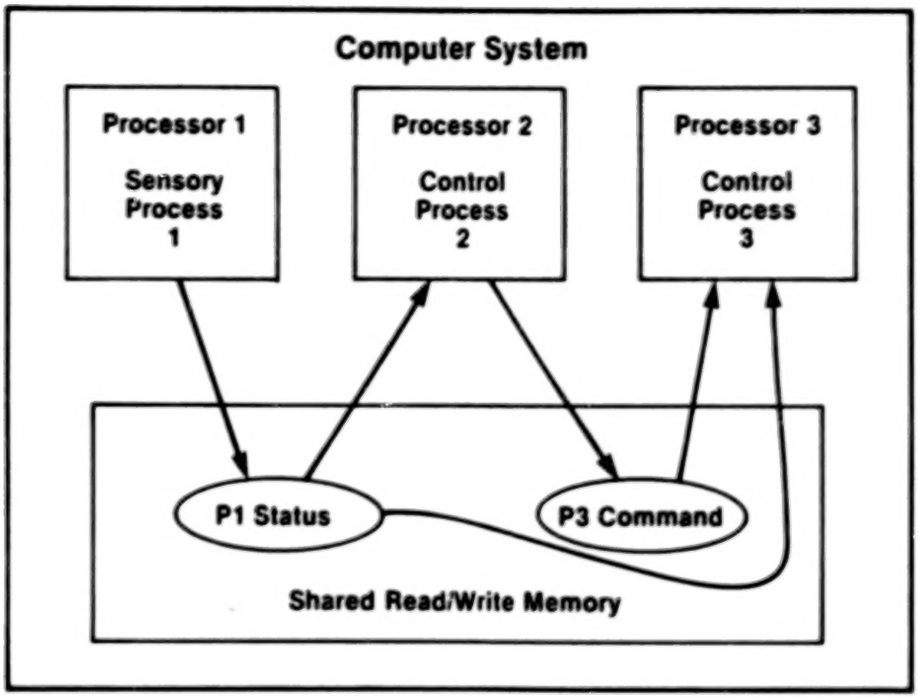


Figure 3.- Memory mapped database.

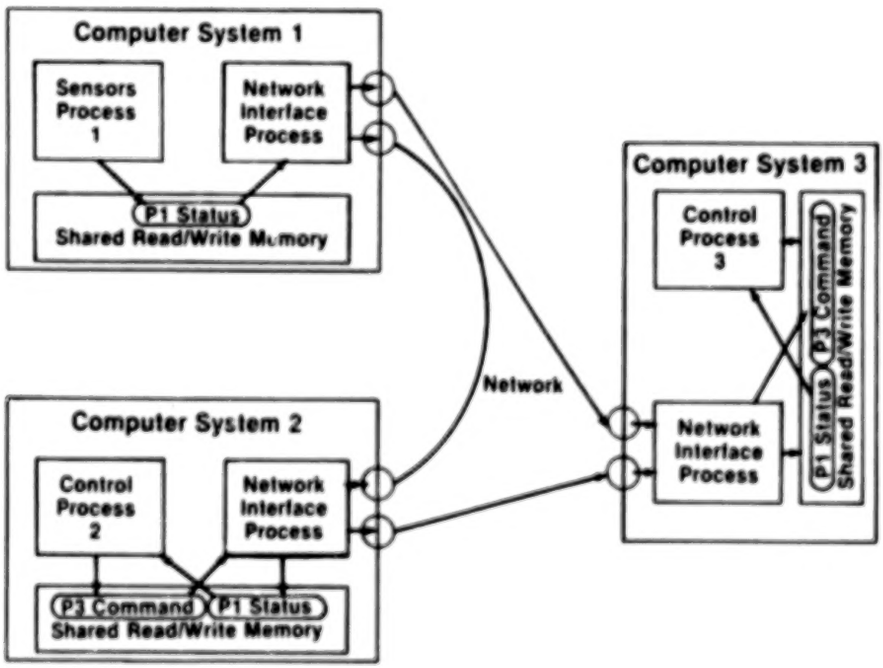


Figure 4.- Distributed memory mapped database.

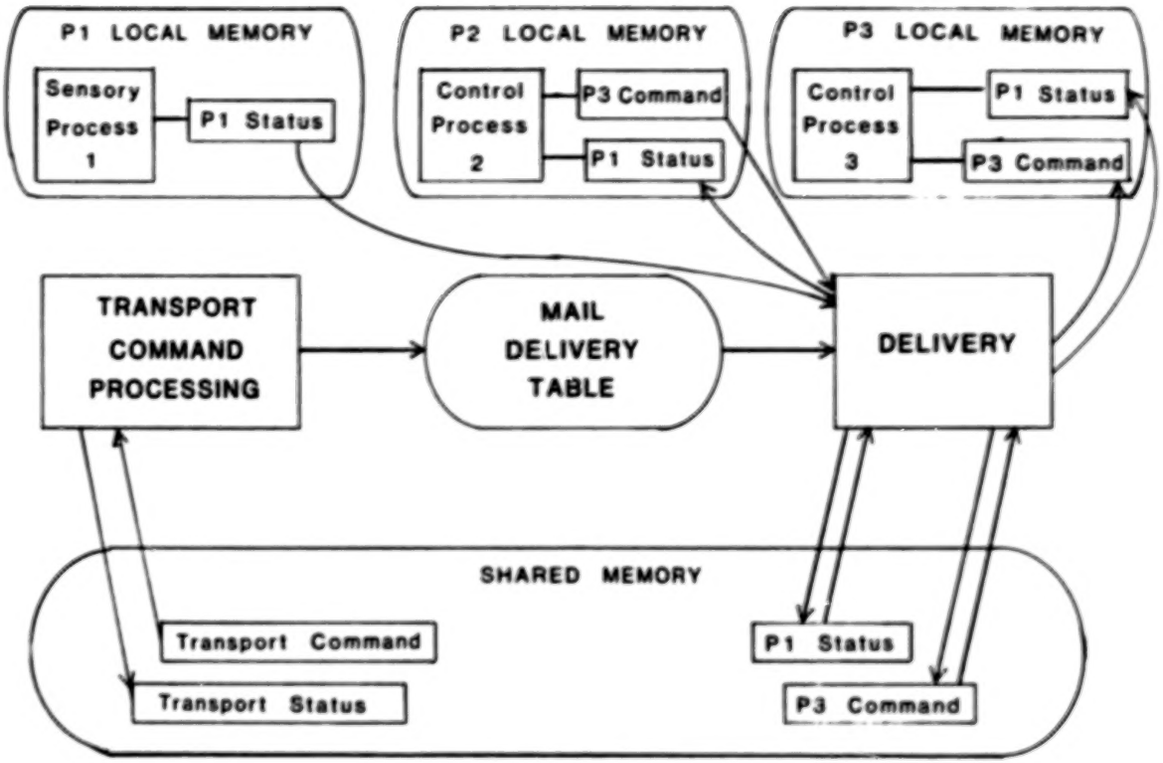


Figure 5.- Local mail transport process.

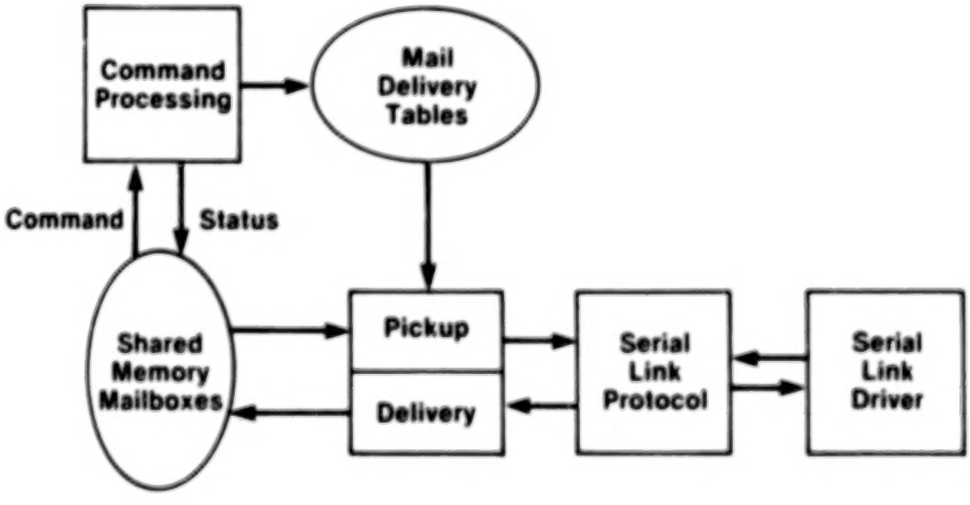


Figure 6.- Network interface process.

D17

THE DATABASE MANAGEMENT SYSTEM:

A TOPIC AND A TOOL

Otho R. Plummer
University of Missouri-Rolla

SUMMARY

Data structures and database management systems are central topics in university computer science curricula, and are common tools employed to deal with the administrative information of a university. It is becoming clear that an understanding of these topics is needed by a much wider audience, ranging from those interested in computer-aided design and manufacturing to those using microcomputers. Moreover, these tools are becoming increasingly valuable to academic programs as they develop comprehensive computer support systems. The wide use of these tools relies upon the relational data model as a foundation. Experience with the use of the IPAD RIM5.0 program and at the University of Missouri-Rolla will be described.

INTRODUCTION

If computer science has acquired special intellectual subject matter which distinguishes it from other disciplines, then this subject matter must include the studies of algorithms and of data structures. Emphasis on the algorithm was evident from the very beginning. The study of data representation struggled out of two demands: initially, how to efficiently store and retrieve bits of information from a physical medium such as a disk drive, and later to maintain logical integrity among related data items. A key observation was that the logical relationships became exceedingly difficult to manage if one was working directly with the physical representation.

It was in the effort to separate the logical structure possessed by a set of data items from the physical means used to record the data that the notion of the database management systems had its inception. The DBMS was to handle storage and retrieval while providing a logical view of the data to the user or his program. It could be thought of as facilitating data handling in somewhat the same way that a language such as FORTRAN or PASCAL facilitates the construction of algorithms.

There have been serious intellectual difficulties associated with data management in all but the simplest cases. Part of the difficulty lies in the dynamic nature of real databases - we correct mistakes, change our mind, or add new items. But before we even get that far, the challenge of how to organize our thinking about the data is often daunting and error prone. Basic principles are required about how to proceed.

The early impetus came from large databases, such as student information, accounting, or inventory. Need for efficient machine utilization imposed many constraints, and it is fair to say that the early results were often less than had been hoped for. It is true that there is hardly a company or university that does not generate its payroll by computer. But there is hardly a manager that is entirely happy with his data processing department, or a data processing department that is entirely happy with the environment in which it must process the data for the organization.

The early situation, if not entirely satisfactory, at least possessed a sort of equilibrium. A known range of problems was being addressed effectively, and data processing had become essential to most larger organizations. This equilibrium has proved short lived. A number of new challenges are at hand that require data management for their solutions.

NEW CHALLENGES FOR DATA MANAGEMENT

The character of the new demands for data management can be conveyed through several examples.

Certainly IPAD participants will be aware of the challenge of dealing with engineering data in a comprehensive and systematic way. One aspect of this need is the design and manufacturing database itself. Another is to provide application interfaces to the data. A third is to permit design and manufacturing information to be used as a corporate resource throughout an industrial enterprise.

This latter need is part of the larger problem of turning raw data into usable management information. The terminology of "Management Information Systems" has given way in many quarters to "Information Resource Management" or to the concept of "Decision Support Systems." The latter is an attempt to focus on the purpose of this activity - to support better decisions - as well as to convey the dynamic nature of the environment in which it is to exist. It is presumed that the end user is the driving force on the database in real time, and that the tools to support this activity must be provided.

That the end user wants to be the driving force can be considered proved by the microcomputer explosion. The real benefits that can be obtained by distributing computing to the desktop seems to be only a small part of the motivation for this phenomenon. More significant is the user's belief that computing would be more valuable if only it were not for the rigidity (or wrongheadedness) of central DP shops and the large database environment. That this view is so common and is accepted so uncritically is clear evidence that users yearn for a different level of computer service.

In actual fact the microcomputer has more potential for producing failed expectations, user frustration, and outright mischief than any other development in computing history. The benefits of automating and extending "yellow-pad" type work are real, large, manageable and positive. But where the hope is to go beyond this into more comprehensive decision support or data management activities, disappointment is very likely. Data organization problems do not get smaller

because the machine gets smaller. The danger is that thousands of good managers will turn into bad systems designers, doomed to repeat all of the mistakes of the past 20 years of data processing.

One final example is provided university academic departments trying to meaningfully integrate the computer into their curricula. To share data and programs among faculty and to let students build up a meaningful collection of software tools virtually mandates a standard, convenient data interface that all programs and users can employ. Just to be relieved of the need to know data formats would be a major boon. Even better is a more comprehensive data management system that can permit output of various programs to be edited, merged and used as input to subsequent processing. A system like the RIM5.0 application interface is of great value in this context.

Apart from the requirement for excellent database management facilities, these examples all have in common the need for end user interaction with the data. This imposes an enormous additional requirement on the computing environment. Somehow the manager or professor must be educated to principles for the structuring of databases and provided tools to implement these principles. Programming is not the sole province of computer professionals. There is a computer literate public that knows something about algorithm construction and knows languages like FORTRAN. Much smaller is the public that knows about data organization or data manipulation languages, but the need for this knowledge is, if anything, greater.

USER EDUCATION AND THE RELATIONAL DATA MODEL

One recently developed approach to database organization and information retrieval is the relational data model pioneered by E.F. Codd. This approach is characterized by a conceptual simplicity, some formal guiding principles, and powerful manipulation tools. Experience at the University of Missouri-Rolla suggests that this framework is usable by a wide segment of the computer public and that the IPAD RIM5.0 program can be a satisfactory vehicle for dissiminating the required education.

In the relational model, data is thought of as being arranged in tables called relations. Each row provides a collection of data about a particular object, data of a given type being placed in a particular column. For instance, one might have a relation SIZE with columns for name, height in inches and weight in pounds:

SIZE : name, height, weight

The specific rows might be:

BOB	72	183
JAN	62	112
JOE	59	84

Another relation AGE might be:

AGE : name, age

with rows

BOB	37
JOE	13

The relational model provides various algebraic operations for combining, subsetting, and otherwise manipulating relations. For instance the "JOIN" operation could put size and age together to provide more complete information for all names appearing in both relations. A command like "JOIN SIZE AND AGE OVER NAME" would produce a relation with the rows

BOB	72	183	37
JOE	59	84	13

Of equal importance are the principles called normalization rules that guide the proper construction of relations. While details of normalization cannot be discussed here, we will give two examples of the use of the relational approach.

Example 1. Student Geometry Problem.

A student project in interactive computer graphics involves setting up a strategy for arranging the basic data in a convenient way. An example problem might be to set up a system for two dimensional geometry with the following properties:

1) support points, lines, polygons, and polygonal curves as basic geometrical entities;

2) permit lines to be defined from points, polygons from lines, and objects from any collection of entities;

3) permit efficient display techniques; and

4) address various connectivity issues such as when two polygons share a vertex or edge.

There are many approaches available for structuring the data. One obvious idea that many students will have is to set up a file for each entity type. Thus there will be relations such as:

POINT : ID, x, y

LINE : ID, point id, point id

TRIANGLE : ID, line id, line id, line id

These relations would seem to parallel the commands by which a user would build up a picture. The user who wanted to make a triangle would issue a "triangle command" the end result of which would be to add a row to the relation TRIANGLE. This plan also retains the interrelationship among the geometrical entities. If a point is moved, only one POINT entry needs to be changed; the lines that use this point automatically reflect the new position.

Difficulties with this approach might begin to crop up with the polygonal lines. Since these lines could contain many line segments, the number of relations would be large. And since these lines would often grow in length as the image develops, there would be complexities involved in moving a line from the relation that handles ten segments, say, to that for eleven segments.

Another class of difficulties might be attendant in deletions. If a line is deleted, for example, the system developer would need to program a search over all rows and columns to determine the affected entities.

Another approach might be simpler if one were building from a relational DBMS such as IPAD's RIM5.0. There might be a special relation

POINT : id, (x,y)

and then a second relation

ENTITY : id, type, component id

For each point there will be one row in the ENTITY relation as well as the POINT relation. When the entity type is "line", there will be two rows in ENTITY, etc. For example, ENTITY could have the rows

1 point]

2 point 2

3 line 1

4 line 2

for a figure containing two points which are connected by a line. This form permits many geometrical facts to be expressed in terms of the relational algebra. For instance, if entity k is a line, all entities which have the line in common can be produced as follows:

1) Define a temporary relation TEMP containing one row and one column and the entry k.

2) JOIN TEMP with ENTITY.

Example 2. Personnel File

A personnel file actually in use at a university was laid out in tabular form with one row per employee. Part of the row contained standard information such as position number, name, salary, department, and title. However, the major purpose of the file was to track the funding sources for each employee, and although most employees were paid from a single account, some individuals were funded from multiple sources. For example, a professor might be funded partly on a department account and partly by a grant. A department chairperson might be funded from one account for the academic portion of the appointment and from another account for the administrative portion. The title would differ also, being "professor" in the first account and "chair" in the second. Appointments could also be spread over multiple departments and have different beginning and ending dates, etc. It was deemed necessary to allow for eight funding sources per person, necessitating a record with the following structure:

pos #, name, acct 1, amt 1, title 1, dept 1, acct 2, amt 2 ...

(A few details have been suppressed. The actual records had 99 fields each.)

Although each record gave a complete description of an employee, a number of drawbacks in the scheme are apparent. Reports were extremely difficult to generate. To even total all salary from a given account required searching over many fields, most of which were empty, following a logic which was in general too complex for interactive work. Maintenance chores were also extensvie. For instance, the entire database had to be searched to move an account from one department to another.

Using normalization principles in a relational database permitted three basic relations to be identified:

EMPLOYEE : pos #, name, salary

FUND : pos #, acct #, title, amount

DEPT : acct #, department

In this framework, FUND carries the critical information. FUND contains one new per funding source for each employee. Projections, intersections and joints permit needed information to be assembled ad hoc and interactively; totals, more formal reports and changes are greatly facilitated as well. The logical simplification achieved is readily apparent.

CONCLUSIONS

Experience to date at the University of Missouri-Rolla suggests the need to extend computing literacy to database principles and knowledge of database tools. We believe that short-course training of broad segments of the user public in these methods is both possible and essential. Previous training in the agorithmic aspects of computing is not a necessary prerequisite for such instruction, nor is

it in any sense a substitute for knowledge of database methods. In addition, we believe that these topics must occur in the university curriculum for a much larger class of student than only computer science majors.

The possibility of carrying out this program successfully depends upon using the relational model as the theoretical underpinning, and upon having available effective relational database management systems for both training and use. The IPAD RIM5.0 program has proved extremely valuable in our efforts made thus far to address these important issues.

THE ROLE OF DBMS IN DESIGN RESEARCH

Steven J. Fenves
Carnegie-Mellon University

SUMMARY

Research in integrated design systems is almost invariably predicated on the existence of a data base that acts as a common repository of data representing the emerging specification of the system or artifact being designed.

First, the representation and processing of detailed constraints is presented, including a dynamic mechanism for activating constraints as a design is "firmed up." Second, a concept of treating multiple sets of constraints (e.g., those arising from building codes of different jurisdictions) as data residing in the data base is presented. Finally, exploratory work on the interaction of DBMS with knowledge-based expert design systems is given.

INTRODUCTION

There is considerable interest and activity in the development of engineering design databases. Such databases are viewed as the primary integration mechanism between the various design processes.

A major aspect of engineering design involves the evaluation and satisfaction of constraints. Constraints arise from many sources: they may represent functional relations (voltage equals current times resistance), they may be externally imposed by codes and standards governing acceptability, or they may represent design objectives or the designer's particular design "style." The ability to represent and process a wide variety of such constraints is a necessary ingredient of any engineering design database. This is especially true in databases integrating several design processes, where the DBMS must serve not as a passive repository of data, but as an active agent performing many of the consistency and integrity checks that are currently done manually.

The major part of the paper deals with a proposal mechanism for representing and processing design constraints. The mechanism can be used for checking constraints, i.e., determining whether they are satisfied or violated, as well as for assigning attribute values such that the applicable constraints are thereby satisfied. Furthermore, the mechanism provides flexibility in sequencing the enforcement of constraints, by allowing new constraints to be applied to a pre-existing state of the database as well as to all subsequent transactions on the database. In both of these respects, the mechanism proposed appears to have applications beyond engineering design.

Extensions of the mechanism to multiple sets of constraints and to knowledge-based expert systems are briefly described.

THE RELATIONAL MODEL

This section presents a brief overview of the salient features of the relational model, to serve as an introduction to the incorporation of constraints in the model.

The Structure of Relations

A relational model is a single level model consisting of a collection of relations represented in two dimensional tabular form [8]. The rows of a relation are called tuples and its columns are called attributes. Attribute values are drawn from an underlying domain which represents all legal uses of instances of the domain. Each tuple represents an entity and contains an attribute value from each domain. All tuples are distinct; duplicates are not permitted. Tuples and attributes have no order; they may be arbitrarily interchanged without changing the data content and meaning of the relation. Tuples are accessed by means of a key, a single attribute or a combination of attributes that uniquely identifies one tuple.

Representation of Relations

Throughout this paper a standard shorthand notation will be used to represent relations. The form of the notation is as follows:

RELATIONname (ATTRIBUTE1name, ATTRIBUTE2name, . . .).

To use a specific example, a database supporting architectural space planning may contain the relation:

ROOMS (roomID, area, breadth, width).

The name of the relation is listed first followed in parentheses by the names of all of its attributes. The underlined attribute of a relation is the key.

Functional Dependence and Normalization

Normalization is the process of removing intrarelation dependencies among attributes of relations. The ROOMS relation introduced above can be used to illustrate functional dependence. The roomID attribute is the key of the relation. For each roomID, there is precisely one corresponding area, breadth, and width value. Each of the attributes area, breadth, and width are therefore functionally dependent on, and functionally determined by, the roomID.

The ROOMS relation also contains a transitive dependency. A transitive dependency is a relationship between non-key attributes, i.e., there exists an attribute that depends on one

or more attributes other than the key. In the ROOMS relation, a room's area is dependent not only on the roomID but on the remaining non-key attributes as well, since area = breadth * width. This dependency could be removed by normalization and the ROOMS relation replaced by two new relations as follows:

ROOMA (roomID, area)
ROOMBW (area, breadth, width).

It is to be noted that this form of normalization assumes that there is only one particular combination of breadth and width which yields a given room area, which may not correspond to the designer's intent.

Integrity Constraints

All relational DBMS's support single attribute integrity constraints which delimit the domain of legal attribute values. For example, in the ROOMS relation one may wish to impose

RULE area IN rooms > 0.
RULE breadth IN rooms > 0.
RULE width IN rooms > 0.

In [3], Fagin suggests that a broader class of integrity constraints which should be enforced by a relational DBMS is the single-tuple constraint:

$$\phi(t) \qquad (1)$$

where ϕ is a constraint and t is an arbitrary tuple of relation R. A tuple t is said to satisfy constraint ϕ, and ϕ is said to hold for tuple t, if $\phi(t)$ is true [3]. Otherwise t does not satisfy ϕ. The purpose of this paper is to formalize this class and suggest mechanisms for their enforcement.

Extensions of integrity constraints referring to the semantic integrity of the data have been widely discussed in the DBMS literature [9], [1]. The constraints dealt with in this paper may be viewed as a special form of semantic integrity constraints, where the semantics or meaning of the data relate to the requirements of engineering design.

DESIGN CONSTRAINTS

Engineering design deals with the evaluation and satisfaction of many constraints. A design is considered satisfactory if it satisfies all applicable constraints.

In this paper, we restrict our attention to single-tuple constraints of the form given in (1).

Canonic Forms Of Constraints

Using the terminology of mathematical optimization, the two canonic forms of constraints are:

$$h(a_i) = 0, \text{ and} \quad (2)$$
$$g(a_i) \leq 0 \quad (3)$$

where the a_i are attributes of a relation R ($i = 1 \ldots k$; k = arity of R) and h and g represent, respectively, equality and inequality functional dependencies among the attributes a_i.

These constraints can be illustrated using the ROOMS relation introduced above. The equality constraint discussed above is:

$$h(\text{area, breadth, width}) = \text{area} - \text{breadth} * \text{width} = 0. \quad (4)$$

The designer's preference or "style" may dictate the following inequality constraints on the aspect ratio of the rooms:

$$g_1(\text{breadth, width}) = \text{breadth} - 2 * \text{width} \leq 0. \quad (5)$$
$$g_2(\text{breadth, width}) = \text{width} - 2 * \text{breadth} \leq 0. \quad (6)$$

These two canonic forms of constraints will be referred to as checking constraints. They simply check the relationship between the attributes of the expression. Their evaluation results in a boolean value of true or false, which can be interpreted as satisfied or violated, respectively. To perform the evaluation, values must be known for all of the attributes in the constraint.

Assignment Using Constraints

For design purposes, it is frequently necessary to recast or paraphrase constraints into an assignment expression:

$$a_j := h(a_i) \quad (7)$$

where ":=" is the assignment operator.

We assume that the original constraint $h(a_i) = 0$ can be explicitly solved for any a_j, that is, $i \neq j$ on the right-hand side (RHS) of expression (7). Thus the expression allows for the assignment of a value to one attribute as a function of the values of the remaining attributes consistent with the constraint. Implicit expressions, which include a_j on the RHS, are not considered in this paper.

Extended Functional Dependence

It can be seen from equation (7) that a_j is functionally dependent on the arguments a_i, $i \neq j$, with respect to the function h. Furthermore, since each of the attributes a_i in the original constraint (2) can be similarly expressed as a function of the remaining attributes, we say that the attributes a_i are fully functionally interdependent with respect to the constraint $h(a_i) = 0$. Similar dependencies occur with the inequality constraint $g(a_i) \leq 0$, which can also be paraphrased as an assignment expression.

Ingredients and Dependents

Assignment expressions for any attribute a_j formed from an equality or inequality constraint provide a dynamic use of the functional dependencies among the attributes. Instead of checking the conformance of all existing attribute values as constraints (2) and (3) suggest, assignments of new attribute values can be made in accordance with expression (7). Distinctions can then be made between dependent and ingredient attributes.

Ingredient attributes are all of the attributes required to evaluate an assignment expression. The dependent attribute is the single attribute expressed as a function of the remaining attributes. In expression (7); a_j is the dependent attribute and the a_i are the ingredient attributes.

Rewriting constraint (4) in the form of one of its possible assignment expressions,

$$\text{area} := \text{breadth} * \text{width},$$

area is the dependent attribute of the expression and breadth and width are the ingredient attributes.

STANDARD TREATMENT OF CONSTRAINTS

It is instructive to review the extent to which the standard treatment of intra-relation dependencies, namely normalization, may be used for enforcing design constraints.

Potential Normalization

The distinction between the two canonic forms of constraints is arbitrary, and will be removed in the following. This is especially true if the attributes in the h-form are defined over real domains, in which case the constraint is normally given in the form:

$$g(a_i) = (|h(a_i)| - \epsilon) \leq 0 \qquad (8)$$

where $|\ |$ denotes the absolute value and ϵ is some specified tolerance.

The reason for introducing the h-form is that conceptually the h-form constraint could be enforced through normalization. Normalization removes from a relation all attributes that are not fully functionally dependent on the key.

For any attribute a_j selected from (7), normalization can be achieved by simply projecting that attribute out of the relation. The dependent or "redundant" attribute a_j can then be obtained from (7) using the current values of the ingredient or "independent" attributes a_i ($i \neq j$). Alternatively, the original relation can be split into two normalized relations. However, that approach introduces another assumption or constraint.

The objection to normalization in design applications is that it is frequently problematic which attribute is dependent or independent. Furthermore, as design proceeds, attributes change from ingredient to dependent, and vice versa. Normalization therefore restricts the free flow of the design process.

Continuing with the space planning example, in one situation the designer may first manipulate room areas until some overall constraint on space is tentatively satisfied. He may then proceed to lay out dimensions, adjusting room areas only as necessary. In another scenario, the designer may start with dimensions derived from a fixed grid and then determine the room areas. Thus, at various times, he may wish to use any one of the three possible assignment expressions paraphrased from constraint (4):

```
area    := breadth * width
width   := area / breadth                    (9)
breadth := area / width.
```

Clearly, normalization based on an a-priori selection of a dependent attribute is not a flexible enough mechanism to satisfy these needs. A new mechanism is needed which should also handle the g-form of functional dependency, which cannot be removed by normalization.

Direct Enforcement Of Constraints

A potential solution is to incorporate constraints (2) and (3) directly into the integrity rules enforced by the database, so that transactions (insertions and updates) on tuples satisfying the constraints are accepted while transactions violating them are rejected. This mechanism is a direct extension of the single-attribute integrity rules found in most relational DBMS's, and is essentially the approach taken in enforcing semantic integrity. However, the integrity rules in current relational DBMS's are defined on the schema, and are active as soon as a relation is instantiated.

This mechanism falls short in the flexibility required, in two aspects. First, in the early phases of design, when information is gathered from many sources and tentative solutions are explored, it is not realistic (or even feasible) to require that all constraints pertaining to the completed design be immediately satisfied on the first trial design. Nor is it realistic to assume that all of the data needed to evaluate the constraints will be available. Constraints will automatically be violated if uninitialized values exist for attributes of a tuple. What is needed is a way to record, control, and change the status of each constraint as the design proceeds. Second, this mechanism does not support the assignment of attribute values subject to the applicable constraints.

AUGMENTED RELATIONS

A new type of relation, referred to as an augmented relation is proposed herein.

Representation Of Constraints

The original relation schema is augmented with additional attributes recording the status of the constraints, one for each constraint on the relation, defined as follows:

$$a_k := (h_k(a_i) = 0) \tag{10}$$

or

$$a_k := (g_k(a_i) \leq 0) \tag{11}$$

where the expression on the RHS represent the constraints and the a_k are the new attributes, one for each constraint k.

The additional constraint attributes are referred to as status attributes. The domains of the constraint status attributes are boolean, with the possible values of true and false, interpreted as satisfied and violated, respectively. Thus, for the example previously used, the ROOMS relation is modified to:

ROOMS2 (roomID, area, breadth, width, areaOK, shapeOK)

where the two new constraints monitored by the status attributes are defined by the expressions:

```
areaOK : = abs(area - breadth * width) <= 0.01
shapeOK := (breadth <= 2 * width) AND (width <= 2 * breadth).        (12)
```

These expressions can be coded as boolean constraint functions and associated with the corresponding status attribute. The first constraint expression would be coded in Pascal as:

```
FUNCTION checkarea (area, breadth, width : real) : boolean;
  BEGIN
     checkarea := abs(area - breadth * width) <= 0.01
  END.
```

The function (or, preferably, the linkage between the DBMS and the function) must automatically set the value of the constraint status attribute to false if any one of its arguments (values of ingredient attributes in the defining expression) is undefined, i.e., has the special value of uninitialized. In this manner the proposed mechanism can always directly evaluate the status of the constraint regardless of the availability of data.

We emphasize that each status attribute is fully functionally dependent on the ingredient attributes that are arguments to its constraint function. Thus, instead of enforcing integrity by normalizing out an arbitrary dependent attribute, we monitor the constraint status by introducing an additional functional dependency.

Enforcement Of Constraints On Augmented Relations

The enforcement mechanism is predicated on the typical design sequence of initially selecting trial values for attributes, then successively "firming up" the design by enforcing conformance with a succession of key constraints until eventually the finished design satisfies all constraints. The mechanism also accommodates the frequent situation where a candidate design turns out to have undesirable properties or to be infeasible with respect to some constraints, in which case constraints need to be withdrawn, new trial attribute values selected, and another design iteration initiated.

The proposed enforcement mechanism uses three new DBMS commands. Initially, none of the constraints are active, i.e., all constraint status attributes have their initial value set to false (or violated). The command to apply a constraint to the current state of the database is of the form:

INVOKE <constraint function> ON <relation>

where <constraint function> is the name of the constraint function to be applied. Multiple constraints can be invoked by a single command. This command causes a batch process to be performed, applying the constraint function to each tuple in the relation in turn, and recording the resulting value (true or false) of the constraint status attribute. For the sample function shown, the command:

INVOKE checkarea ON rooms2

causes the assignment

areaOK := checkarea (area, breadth, width)

to be performed for each tuple.

As part of the process, the non-conforming tuples are located and output. This operation is equivalent to a standard query, e.g., for the sample relation ROOMS2:

SELECT FROM rooms2 WHERE (areaOK ≠ true).

Remedial action can then be taken to bring these tuples into conformance.

A relation holds design information about many objects, with each tuple representing one. Because the nature of the design process is such that objects may be designed at varying times and in varying orders it is not always realistic to impose a constraint invocation on the entire relation. Therefore, the INVOKE command can be combined with a SELECT FROM clause, allowing invocation of the constraint only on those tuples satisfying the selection criterion.

The command to apply a constraint on all subsequent transactions is of the form:

ACTIVATE <constraint function> ON <relation>.

The effect of ACTIVATE is equivalent in outcome to a standard integrity rule of the form:

RULE <constraint status attribute> IN <relation> ≠ false.

That is, for each transaction (insertion or update) on a tuple, the function for evaluating the constraint status attribute is invoked for that tuple only and the transaction is accepted if the constraint status attribute evaluates to true or rejected if it evaluates to false.

Conceptually, there is no need to INVOKE a constraint function on the current state of the database before that constraint function is ACTIVATEd for subsequent transactions. However, there is no assurance that every tuple will be involved in a transaction. Therefore, complete integrity of the relation can be assured only if ACTIVATE is preceded by an INVOKE. This can be achieved by having ACTIVATE automatically initiate the corresponding INVOKE and proceed only if all tuples conform.

Finally, a command of the form:

DEACTIVATE <constraint function> ON <relation>

suspends the enforcement process and the relation reverts to its initial mode with respect to the specified constraint(s), allowing modifications to be made without checking. The constraint can be subsequently re-INVOKEd and re-ACTIVATEd.

When constraints can be defined at any time during the life of a database, it must be possible to apply a newly defined constraint on the attributes of a relation on a tuple by tuple basis, not only to all subsequent transactions but also to the pre-existing state of the relation, i.e., to all previous transactions. The three commands described above satisfy these requirements. The INVOKE command brings the status attributes up to date from all previous transactions, while ACTIVATE command insures constraint compliance for all subsequent transactions. The commands also enable the integrity of a relation with respect to a constraint to be restored after the use of that constraint has been suspended by deactivation.

Assignment Of Attributes Satisfying Equality Constraints

Once a mechanism is implemented to invoke a constraint function (either in batch or one transaction at a time) and evaluate and store a constraint status attribute value based on the current values of its ingredient attributes, a major further step can be taken. It was pointed out previously that at various stages of design, all paraphrases $a_j := h(a_i)$, $i \neq j$ obtained from a given equality constraint $h(a_i) = 0$ may prove useful to the designer. That is, he may wish to evaluate and store any attribute value dependent on the current values of the other attributes appearing in the constraint.

This extension can be readily achieved in the proposed constraint enforcement mechanism by defining a set of assignment procedures for each constraint. Each procedure returns values of two attributes: the selected dependent attribute evaluated from $a_j = h(a_i)$; and the constraint status attribute. The latter is automatically assigned the value true (or satisfied), since the former is evaluated in such a way that the constraint expression is satisfied. The reason for requiring that the procedure return two results, i.e. that it be a procedure rather than a function, is that all enforcement control is performed on the constraint status attribute. Thus, the command:

INVOKE ⟨assignment procedure⟩ ON ⟨relation⟩

will invoke the procedure in turn for each tuple, compute and store the value of the dependent attribute, and set the constraint status attribute to true. A subsequent SELECT FROM query to locate violated status attribute tuples would return an empty relation. Similarly once a command of the form:

ACTIVATE ⟨assignment procedure⟩ ON ⟨relation⟩

is given, the same process is performed dynamically for each tuple in the relation affected by the transaction.

Continuing with the example, a procedure to assign the area based on the first expression in (9) would be coded in Pascal as:

```
PROCEDURE setarea (breadth, width : real;
                   Var area : real; Var areaOK : boolean);
  BEGIN
    area   := breadth * width;
    areaOK := true
  END.
```

Three comments are in order. First, the attributes included in h-form equality constraints exhibit full functional interdependence so that relations such as the ROOMS relation are not in normal form, i.e., one attribute is redundant. We have suggested that this redundancy not be removed by normalization. Instead, a mechanism for both constraint enforcement and attribute value assignment is proposed. The reason for proposing the assignment procedure here is that it provides the desired flexibility for design; it also serves as an introduction to assignment procedures based on g-form inequality constraints, which cannot be normalized. Second, the presentation further emphasizes the functional dependence of the constraint status attribute, i.e., its value is known to be true when the assignment procedure is executed. The need for the "redundant" status attribute will become clear when hierarchies of constraints are introduced. Third, the sample assignment procedure shown above was "manually" derived from the original constraint expression (4). In a fully implemented system, all such procedures could be symbolically derived using a symbolic algebra system such as MACSYMA [6].

Assignment Of Attributes Satisfying Inequality Constraints

As pointed out previously, g-form constraints cannot be normalized by conventional means. On the other hand, in defining the constraint status attribute no distinction was made between constraint expressions based on the g- or h-form. The next logical step is to extend the concepts of the previous section to assignments based on inequality or g-form constraints.

The extension sought follows readily, with one fundamental but obvious distinction: whereas any paraphrase $a_j = h'(a_i)$ of a constraint $h(a_i) = 0$ yields an expression for a unique value of a_j, an equivalent paraphrase $a_j \leq g'(a_i)$ of a constraint $g(a_i) \leq 0$ yields only a bound on the possible values of a_j. Therefore, each g-form constraint introduces a bounded functional dependence of a_j on the remaining attributes a_i. Thus, with respect to the g-form constraint, the designer is free to choose any value for the attribute a_j subject to the bound.

There are (at least) two implementation alternatives for incorporating assignment procedures based on g-form constraints in a relational DBMS. If it is intended that the designer exercise his choice within the bounds interactively, the assignment procedure based on constraints (5) and (6) may be of the form:

```
PROCEDURE set breadth(width : real; VAR breadth : real;
                      VAR shapeOK : boolean);
  BEGIN
    breadth := MIN[2 * width, {user chooses}];
    shapeOK := true
  END;

PROCEDURE set width (breadth : real; VAR width : real;
                     VAR shapeOK : boolean);
  BEGIN
    width := MIN[2 * breadth, {user chooses}];
    shapeOK := true
  END.
```

In this implementation, when one of the assignment procedures is INVOKEd, the designer would be requested to choose a value of a_j for each tuple; similarly, after the procedure is ACTIVATEd, he would be asked to choose a value every time a transaction produces new values of the a_i for some tuple.

Alternately, if the designer's logic for choosing a_j is known in advance, that logic can be directly incorporated in the assignment procedure and the resulting procedure treated in exactly the same way as for the h-form. This alternative would likely be implemented at the detailed levels of any design activity, where one typically chooses a value just satisfying a g-constraint.

Extension To Non-Numeric Constraints

The preceding presentation dealt with constraints on attributes with numeric domains. It should be clear that the same method can be applied to attributes whose domains are non-numeric. The only requirement is that the constraint status be stated as a boolean expression.

Extending the previous example, assume that the relation ROOMS has two additional attributes: an attribute called function with domain {'public,' 'private'} and an attribute called location with domain {'internal,' 'external'}. The constraint "a public room must have an external location" is expressed in checking form as:

```
FUNCTION usageOK (function, location : string) : boolean;
  BEGIN
   usageOK := NOT ((function = 'public') AND (location = 'internal')).
  END.
```

There are two possible assignment procedures for this constraint. These are:

```
PROCEDURE setfunction (location : string; VAR function : string; VAR usageOK : boolean);
  BEGIN
    IF (location = 'internal')
        THEN function := 'private'
        ELSE function := {user chooses};
    usageOK := true
  END;

  PROCEDURE setlocation (function : string; VAR location : string;
                         VAR usageOK : boolean);
  BEGIN
    IF (function = 'public')
        THEN location := 'external'
        ELSE location := {user chooses};
    usageOK := true
  END.
```

MULTIPLE CONSTRAINTS

In the presentation so far, we have dealt with individual single-tuple constraints, and discussed methods for representing and checking them as well as for assigning attribute values subject to such constraints. The extension of these methods to multiple constraints (still pertaining to single tuples of single relations) involves two issues: the treatment of multiple dependents and the treatment of hierarchies of constraints.

Multiple Dependents

Two constraint status attributes are said to possess shared ingredients if the intersection of the attributes in their defining constraints is not nil. That is, two constraint status attributes of the form shown in (11), say

$$a_{k1} := (h_{k1}(a_i) = 0), \text{ and}$$

$$a_{k2} := (h_{k2}(a_i) = 0)$$

share those ingredients a_i which appear in both h_{k1} and h_{k2}. Conversely, any such shared ingredient has both a_{k1} and a_{k2} as one of its dependents. In the space planning example, breadth and width are shared ingredients of the status attributes areaOK and shapeOK, and both of the latter are dependents of breadth and width.

As long as only constraint status attributes are evaluated using constraint functions, it is immaterial whether a given attribute a_i has multiple dependents or just one. If, however, an assignment procedure is used to assign a value to an attribute a_i possessing multiple dependents, a potential inconsistency arises. If a_i is assigned a value based on satisfying constraint h_{k1}, the status of constraint h_{k2} is potentially affected.

Consistency can be maintained by including in the assignment procedure for attribute a_i additional statements to evaluate the status of all constraints dependent on a_i. To illustrate, assume that the designer wishes to have available two procedures for computing the width of rooms: one based on the area equality constraint and one making the rooms square (thus satisfying the aspect ratio constraint). The following two functions and two procedures are then needed:

```
FUNCTION checkarea (area, breadth, width : real) : boolean;
        BEGIN
           checkarea := abs(area - breadth * width) <= 0.01
        END;

FUNCTION checkshape (breadth, width : real) : boolean;
        BEGIN
           checkshape := (breadth / width <= 2.0) AND
                         (breadth / width >= 0.5)
        END;
```

```
PROCEDURE setwidth1 (area, breadth : real; Var width : real;
                     Var : areaOK, shapeOK : boolean);
        BEGIN
          width   := area / breadth;
          areaOK  := true;
                     {status known, assignment based on this constraint};
          shapeOK := checkshape (breadth, width)
                     {to get status of other dependent of width}
        END;

PROCEDURE setwidth2 (breadth : real; Var width : real;
                     Var : areaOK, shapeOK : boolean);
        BEGIN
          width     := breadth;    {designer's choice}
          shapeOK   := true;       {above satisfies shape constraint}
          areaOK    := checkarea (area, breadth, depth)
        END.
```

The functions checkarea and checkshape can be ACTIVATEd singly or together, as before. If setwidth1 is ACTIVATEd after checkshape and shapeOK evaluate to false, the transaction is rejected; vice versa for setwidth2, checkarea and areaOK.

The extension to more than two dependent constraints is straightforward. We emphasize again that the functions and procedures shown, including the evaluation of "sibling" dependent status attributes, can be symbolically generated from the original constraints.

Hierarchies Of Constraints

Until now, we considered only basic constraint status attributes whose ingredients were attributes in the original, non-augmented relation. It is natural to consider additional, "higher level" constraint status attributes whose ingredients are "lower level" status attributes, forming a hierarchy of status attributes of arbitrary depth. The design process is complete when the "topmost" constraint status attribute evaluates to true for every tuple.

In the example, the "topmost" status attribute could be defined by the expression:

roomOK := areaOK AND shapeOK. (13)

To insure full consistency with respect to all "lower-level" constraints, the constraint functions corresponding to the "higher-level" status attributes must explicitly evaluate the "lower-level" constraint status attributes. Thus, the function corresponding to (13) is:

```
FUNCTION checkroom (area, breadth, width : real): boolean;
    Var areaOK, shapeOK : boolean);
    BEGIN
      areaOK    := checkarea (area, breadth, width);
      shapeOK   := checkshape (breadth, width);
      checkroom := areaOK AND shapeOK
    END.
```

Assignment Using Multiple Constraints

Assignment procedures based on simultaneous satisfaction of multiple constraints are possible. Holtz [5] presented a method for the symbolic manipulation of design constraints, including multiple equality and inequality constraints. In Holtz's original work, numerical values are given to some of the ingredient attributes, and bounds are generated for the remaining dependent attribute(s). It is possible to use Holtz's program with all attributes given symbolically and generate the paraphrased assignment expression for a selected dependent attribute, subject to an overall constraint.

The main difference between assignments based on single and multiple constraints is that in the latter it cannot be taken for granted that the constraints can all be satisfied simultaneously. That is, the ingredient attribute values may be such that no feasible assignment can be made to the dependent attribute. Thus, assignment procedures based on multiple constraints must be written and processed so that they return either an attribute value together with overall constraint status value of true, or a status value of false.

CONCLUSIONS AND EXTENSIONS

The objective of this paper was to present a method for handling a broad class of single-relation, single-tuple constraints typical in engineering design applications. Instead of relying on normalization - where normalization is possible at all - to remove functional dependencies, the approach presented introduces additional attributes representing the status (satisfied or violated) of each constraint thereby increasing the functional dependence of the relation.

A direct consequence of this approach is that passive constraint checking can be readily extended to active assignment of attribute values that automatically satisfy constraints. In addition, a flexible constraint enforcement mechanism permitting dynamic control is achieved.

A prototype system implementing many of the components presented has been programmed in Pascal [7]. Some of these components are currently implementable as an extension of the "computed" attribute function of RIMS [2].

Multi-Relation Constraints

The next logical step is to extend the method to multi-tuple and multi-relation constraints. We can only offer preliminary thoughts on such an extension. The critical question is the following: given a constraint evaluation function or constraint-based assignment procedure,

where are their arguments (the attributes needed for the evaluation) to be found in the database? In this paper, it is assumed that all needed arguments are attribute values of the current tuple. In the most general case, the arguments may include attribute values from other tuples of the current relation or from other relations. Using the space planning example for the last time, constraints on a room may involve attributes of:

1. the city, building, floor, etc. "owning" or containing this room;
2. the walls, equipment, people, etc. "owned" by or contained in the room; or
3. adjacent (or otherwise related) rooms.

It appears that the proposed method can be extended to such cases, provided that the functions and procedures themselves have access to the DBMS and can use PROJECT, SELECT, and JOIN, as well as aggregate operations on relations (e.g., to evaluate constraints such as "room area ≥ sum of areas of equipment in room"). The issues of implementation efficiency are the same as those discussed in [1]. However, there is a serious restriction on the attributes for which assignment functions can be used: only attributes of the "owning" entity may be assigned values based on aggregates of "owned" attributes; the reverse is not possible.

Multiple Sets of Constraints

Frequently, design organizations have to be able to operate under multiple sets of constraints. This is particularly true in the building and construction industries, with their multiplicity of state and local building codes and specifications. Each of these codes or specifications can be expressed as a separate set of constraint consistency checking functions.

An engineering design DBMS could eventually accept as part of the schema definition the name of the set of consistency checking functions to be applied, so that the appropriate executable file of functions is linked to the schema. In this fashion, multiple sets of constraints could be readily treated as "executable data". As before, there would be difficulties of converting passive constraint checking functions into active assignment procedures for each set of constraints, unless the conversion can be performed symbolically.

Knowledge-Based Constraints

There has been considerable interest and activity in combining DBMS's with knowledge-based expert systems (KBES). Preliminary studies indicate that the proposed mechanism may be used for interfacing DBMS's and KBES's in two different control modes.

In one mode, the KBES is "in control" but uses the DBMS to obtain appropriate attribute values. Thus, in a typical rule-based KBES, executing a rule of the form

IF⟨condition⟩ THEN⟨action⟩

would require DBMS accesses to establish the value (true or false) of the ⟨condition⟩. Similarly, the result of the ⟨action⟩ may be to place some new attribute value or values in the database. In the terminology of this paper, each such rule is equivalent to an assignment procedure for the derived attribute(s) of the ⟨action⟩ part in terms of its ingredient attributes appearing in the ⟨condition⟩ part.

Conversely, the DBMS may be "in control", but instead of a single algorithmic assignment procedure for a given attribute, there may be several heuristic rules for the assignment based on different ⟨conditions⟩. In this case, the assignment procedure could be replaced by a rule set to be processed by the KBES. Since the rule set in a KBES is typically not complete, the constraint status variable must be set to "unsatisfiable" if none of the ⟨conditions⟩ in the rule set can be satisified, i.e., if there is no applicable rule in the set.

ACKNOWLEDGEMENTS

Portions of this work were sponsored by the National Science Foundation under grant MCS7822328, entitled "Data Base Methods for Design." This paper is based largely on an internal report co-authored by Dr. William J. Rasdorf [4].

References

[1] Bernstein, P., Blaustein, B. and Clarke, E.M.
Fast Maintenance of Semantic Integrity Assertions Using Redundant Aggregate Data.
Proceedings of the Sixth International Conference on Very Large Data Bases, Montreal, Canada, October 1-3, 1980.

[2] Erickson, W.J., Gray, F.P. and Limbach, G.
Relational Information Management System.
Version 5.0 edition, Boeing Commercial Airplane Company, Seattle, WA, 1981.

[3] Fagin, R.
A Normal Form for Relational Databases That is Based on Domains and Keys.
ACM Transactions on Database Systems 6(3):387-415, September, 1981.

[4] Fenves, S.J. and Rasdorf, W.J.
Treatment of Engineering Design Constraints in a Relational Database.
Technical Report DRC-12-14-82, Design Research Center, Carnegie-Mellon University, Pittsburgh, PA, December, 1982.

[5] Holtz, N. M.
Symbolic Manipulation of Design Constraints: An Aid to Consistency Management.
PhD thesis, Carnegie-Mellon University, Pittsburgh, PA, May, 1982.

[6] Mathlab Group.
MACSYMA Reference Manual, Version Nine.
Technical Report, Laboratory for Computer Science, 1977.

[7] Rasdorf, W. J.
Structure and Integrity of a Structural Engineering Design Database.
Technical Report DRC-02-14-82, Design Research Center, Carnegie-Mellon University, Pittsburgh, PA, April, 1982.

[8] Sandberg, G.
A Primer on Relational Database Concepts.
IBM Systems Journal 20(1):23-40, 1981.

[9] Stonebraker, M.
Implementation of Integrity Constraints and Views by Query Modification.
In *Proceedings of the 1975 SIGMOD Workshop on Management of Data*, pages 65-78. Association for Computing Machinery, New York, NY, 1975.

THE IMPACT OF IPAD ON CAD/CAM DATABASE UNIVERSITY RESEARCH

Lanse M. Leach
United States Military Academy

Michael J. Wozny
Rensselaer Polytechnic Institute

SUMMARY

NASA's IPAD program has provided direction, focus, and software products which have significantly impacted on CAD/CAM database research at Rensselaer Polytechnic Institute and follow-on research at the United States Military Academy. The relationship of IPAD to the research projects involving the storage of geometric data in common database facilities such as database machines, the exchange of data between heterogeneous databases, the development of IGES processors, the migration of large CAD/CAM database management systems to non-compatible hosts, and the value of RIM as a research tool is described.

INTRODUCTION

In 1980, the Rensselaer Polytechnic Institute's Center for Interactive Computer Graphics initiated a CAD/CAM database research program to develop new methods for the common storage and exchange of geometric data as part of the design and manufacture processes. Initial direction and focus to the research activities were gained from participation as a observer at IPAD Technical Advisory Board meetings.

IPAD MIGRATION TO THE IBM ENVIRONMENT

Use of the IPAD Information Processor (IPIP) was planned as the first step in the research effort. Without any Control Data Corporation hardware, an investigation into the effort required to migrate the IPIP software from CDC to IBM was undertaken. The most difficult part of the task was thought to be rewriting that portion of IPIP which was coded in CDC machine language. However after three months effort, the IPIP File Control System, which accounted for the majority of the machine language code, was converted and tested in the IBM environment. The difficult part of the effort turned out to be the migration of the PASCAL code. Even though PASCAL is a relatively

portable language, the specific use of the sixty bit CDC word length made the code migration generally impossible. After several months effort it became obvious that up to ten man-years would be required for the migration of sufficient code to create a usable system. At the same time, Boeing was making considerable changes and enhancements to the base code, making RPI's base version obsolete.

Because of the revised workload estimates and the obsolete base code, the migration project was terminated. The time spent on the migration was considered very worthwhile since considerable insight into the design of IPIP and the requirements for a large-scale CAD/CAM database management system was gained by the research team (ref. 1).

EXCHANGE OF DATA BETWEEN HETEROGENEOUS DATABASES

With large databases such as those being developed in the IPAD program, a requirement to move data between databases and application systems became evident. This led to a major research project in the exchange of data between databases. The research resulted in the development of a general method to facilitate exchange of geometric and related data between heterogeneous CAD/CAM databases with minimal or no knowledge of the database facility. The system includes (1) a language, Common Database Interface Language (CDIL), to describe control and communication parameters, object, entity, and property descriptions, an entity mapping table, and entity translation processes, (2) a virtual database facility interface composed of database management system dependent functions to retrieve, store, link, unlink, and delete objects, entities, and properties, and (3) a run-time system that performs flow control and dynamic entity and memory management for the exchange of data between multiple common database facilities.

This method for data exchange has the advantages of flexibility, expansibility, portability, and efficiency over current pre- and post-processor methods of data exchange. Use of the language interface provides the missing link between the development of common database facilities such as NASA's IPAD and the development of exchange formats such as the Initial Graphics Exchange Specification (IGES) for data exchange between CAD/CAM databases, systems, and application programs. In the prototype development, NASA's RIM was used as the data manager for the common CAD/CAM database used in the testing of the system (ref. 2).

DATABASE MACHINE INTERFACES FOR CAD/CAM

The support of the relational data model in the IPAD Information Processor and the availability of the Relational Information Manager (RIM) assisted in the research investigation of how CAD/CAM geometry

can best be stored using the relational data model. With the acquisition of a Britton-Lee Intelligent Database Machine (IDM), interfaces were developed for application programs and interactive terminals to support this investigation. This continues to be an active research project at RPI (refs. 3 and 4).

EXCHANGE OF CAD/CAM DATA WITH IGES

With the growing acceptance of IGES as the primary format for data exchange, an investigation was conducted in the design considerations for selected geometric, annotative, and associativity entities. Full IGES implementation will be required in any large scale database management system in the support of CAD/CAM such as IPAD (ref. 5)..

CONCLUDING REMARKS

For the last three years the IPAD has provided direction and focus to the CAD/CAM database research activities at Rensselaer Polytechnic Institute. As IPAD continues its leadership in solving the difficult and complex problems of the storage of CAD/CAM data in the distributed and networked environments, this direction and focus will continue. CAD/CAM database research remains one of the major research programs in the Center for Interactive Computer Graphics at Rensselaer Polytechnic Institute. The data exchange research effort is being continued by the first author at the United States Military Academy.

REFERENCES

1. Owens, Alan B.: Implementation of the IPAD File Control System on the IBM CMS Operating System. Rensselaer Polytechnic Institute, May 1982.

2. Leach, Lanse M.: A Language Interface for Data Exchange Between Heterogeneous CAD/CAM Databases. Ph.D. Thesis, Rensselaer Polytechnic Institute, April 1983.

3. Ho, Peter: Implementation of the Intelligent Database Language on the Prime 750 for the IDM 500. M.S. Thesis, Rensselaer Polytechnic Institute, May 1983.

4. Schonfeld, Craig: A Programmers Interface to the IDM-500. Rensselaer Polytechnic Institute, May 1983.

5. Montine, James L.: IGES Processor Design Considerations for Selected Geometric, Annotative, and Associativity Entities. Rensselaer Polytechnic Institute, May 1983.

END

DATE

FILMED

JUN 19 1984